Alexandre Moura Antunes Dias

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Channel Selection Algorithms for Cognitive Radio Systems

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Orientador: Prof. Dr. Rodolfo Alexandre Duarte Oliveira,
Prof. Auxiliar com Agregação,
Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa

Júri

Presidente:	Prof. Dr. Luís Augusto Bica Gomes de Oliveira, Prof. Auxiliar Agreg. FCT/UNL
Arguente:	Prof. Dr. Pedro Miguel Figueiredo Amaral, Prof. Auxiliar FCT/UNL
Vogal:	Prof. Dr. Rodolfo Alexandre Duarte Oliveira, Prof. Auxiliar Agreg. FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

março, 2019

Channel Selection Algorithms for Cognitive Radio Systems

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Everybody is a genius, but if you judge a fish by it's ability to climb a tree, it will live it's whole life believing it is stupid.

Albert Einstein

Acknowledgements

First of all, I would like to express my sincere thanks to my adviser, Professor Rodolfo Alexandre Duarte Oliveira for all the availability, commitment, motivation and assistance during the development of the dissertation.

I would also like to thank to the Faculty of Science and Technology of the University Nova de Lisboa, as well as, all the teachers I had the opportunity to contact with at the transmitted knowledge, which enabled me to develop my skills and complete this step successfully. I also acknowledge the support of Instituto de Telecomunicações under the Project CoSHARE (LISBOA-01-0145-FEDER-0307095 - PTDC/EEI-TEL/30709/2017), funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Programa Operacional Regional LISBOA (LISBOA2020), and by national funds, through Fundação para a Ciência e Tecnologia (FCT).

I want to thank my family, especially my mother and sister, for all the love, affection, unconditional support they gave me so that I could achieve all of my goals. I also want to thank my father that, although he is no longer present, he taught me a lot and always gave me the strength which made me the person I have become. Special thanks to my grandparents that even though they were not here, they were essential to me. I would also like to thank the Martins family and, especially, my godfather of chrism Marco Martins for making us part of the family and for always helping us in good and bad times.

This academic journey was only possible thanks to my classmates João Morgado, Carolina Lagartinho, Pedro Guerreiro, Bruno Santos, Beatriz Salvado and David Alexandre, who helped and supported me in my academic life, always encouraging me.

Special thanks should be given to Nuno Pinto for all the experiences and adventures we have shared. Without those experiences we would not be who we are today. Thank you very much.

There is a group of people who for their friendship deserve to be present in these acknowledgements. I will enunciate some such as Sérgio Mendes, Mariana Rodrigues, Tiago Araújo, Rodrigo Mendes, Diogo Martins, César Pinto, Gonçalo Luc among others.

My special thanks are extended to my work team IaC at IBM for all the shared knowledge, friendship and assistence since I started my work life.

I want to thank MJM, Coro 3/4 and my catechesis for opening my eyes to a different way of seeing life and for helping me understand that helping others around me is my life mission. Some people who accompany me in these groups are Bruno Pinto, Diana Ribeiro,

Wilson Cruz, Marta Maciel, André Lopes, Ana Gomes, Rui Marçalo, Vanessa Fernandes among others.

To Inês, for all the affection, support and strength at all times and especially for making my present and future life in a melody of love. Thank you so much for never giving up on me. I would also like to thank Inês' family for all the support and help throughout this dissertation.

Last but not the least, I want to thank God, who has sustained me through these work, the best and toughest years of my entire life, always with me.

Abstract

The Spectrum of Radio Frequency plays a key/central role in supporting the operation of the whole range of mobile wireless devices. Such centrality derives from the fact it is necessary when operating different technologies, such as TV, cellular networks, and satellite transmissions, among others. Hence, the urgent need to design a better management of these technologies as a means of minimizing the amount spectrum that is used.

This study was developed in two different stages. During a preliminary stage we came up with a modelling method of the service time duration in cognitive radio networks, which fosters a better management of the spectrum through its opportunistic use. In a second stage, we propose four algorithms to manage these cognitive radio networks, using the modelling method that was defined in the first stage, as a basis.

After implementing all the simulations, it was possible to verify the feasibility of this modelling methodology and also to confirm the expected results. It should also be pointed out that the four suggested algorithms were tested by carrying out simulations, being effective solutions for difference operational scenarios.

Keywords: Cognitive Radio, Radio frequency, Spectrum, Modeling, Performance Analysis.

Resumo

O espectro de radio frequência desempenha um papel central no suporte de toda a operação dos dispositivos móveis sem fios. Tal centralidade prende-se com o facto de ser necessário para o funcionamento de diferentes tecnologias, tais como a rádio, a TV, as redes celulares e transmissões via satélite, entre outras. Daqui resulta uma necessidade premente de conseguir desenhar uma melhor gestão destas de forma a minimizar a quantidade de espectro utilizado.

Neste estudo, é desenvolvido, numa primeira fase, um método de modelação do tempo de serviço em redes de rádio cognitivas, as quais permitem uma melhor gestão do espectro através da sua utilização oportunistica. Num segundo momento, foram elaborados quatro algoritmos de gestão destas redes, utilizando, como base, o método de modelação definido na primeira fase.

Após a realização de diferentes simulações, foi possível verificar a exequibilidade desta metodologia de modelação, bem como observar os resultados esperados. Acresce referir que os quatro algoritmos propostos também foram testados através de simulações, apresentando soluções específicas para diferentes cenários de trabalho.

Palavras-chave: Redes Cognitivas, Espectro de frequências, Modelação, Avaliação de Desempenho.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The radio frequency spectrum is one of the most required resources for radio communications. It is used in a wide range of fields such as radio and TV broadcasting, cellular communications and satellite communications, among others. In information and communication technologies, the radio spectrum supports the channels in which wireless telecommunications signals are transmitted. Throughout the world, spectrum allocation and planning are essential tools to support licensed and non-licensed services operating with guaranteed levels of quality.

The development of mobile networks has grown exponentially, consuming more and more resources. Almost the entire spectrum bands are already allocated to different services, causing a problem that we need to face in the future of wireless systems: how to find suitable bands to meet the predicted demand for future services. Although the great majority of band is currently allocated to specific services, spectrum occupancy measurements support the argument that large portions of the allocated frequency bands are only partially occupied. This means that multiple spectrum bands are currently underutilized, indicating that new spectrum management policies may provide significant advantages.

To meet the demand for future services, new and more flexible algorithms need to be developed for wireless networks. The technology that will improve the spectrum's usage is called Cognitive Radio. Although Cognitive Radio may have a high potential, it entails several challenges, such as the "availability of channels"in the spectrum and the frequency in which the channel is available. Cognitive radio networks are composed by licensed users (Primary Users - PU's) and non-licensed users (Secondary Users - SU's) who usually can reutilize the spectrum in an oportunistic way.

This dissertation starts by presenting an overview of how Cognitive Radio networks work and how the spectrum utilization may be improved. The work is then oriented to understanding the statistics involved in the time required to opportunistically transmit a packet in a Cognitive Radio network, also known as packet service time. In Chapter 3 we describe the mathematical model behind the estimation algorithm and in Chapter 4 we present four algorithms that can be used by Cognitive Radio networks to select available channels. The motivation underlying the study of the service time is its adoption in Cognitive Radio networks to improve the performance when multiple channels are available.

1.2 Objectives

The first goal of this work is related with a preparatory modeling task capable of computing the service time to transmit a packet when the statistics of the occupancy of the channel by primary users is known. In a second step the objective is to use the model to develop an estimation methodology to determine the service time in real-time when a few samples of the channel's occupancy are available. Finally, a third goal aims to develop different algorithms to perform cognitive radio channel selection according to the required quality of service and channel statistics.

1.3 Contributions

This dissertation seeks to contribute with the development of an estimation method that will pave the way for a more effective use of Cognitive Radio networks. One of the advantages of this approach relies on the fact that we can predict if a channel will be more or less occupied than another. Four algorithms were developed and tested. These algorithms aim to study how different scenarios can be addressed using a Cognitive Radio methodology. To help to promote a further study using this estimation methodology to determine the service time in real time when a few samples of the channel's occupancy are available, we provided all the details and procedures concerning the implementation of the four algorithms. The same applies to the code that was developed, which is listed in the annexes.

CHAPTER 2

Related Work

2.1 Cognitive Radio

Cognitive Radio allows radio transceivers to detect available wireless channels, discovering which communication channels are in use and which are not. The main goal of Cognitive Radio devices is to move into vacant channels while avoiding occupied ones [12]. This allows the transmission of multiple signals into a single channel, optimizing the use of spectrum while minimizing interference with other users.

Cognitive Radio adopts many technologies including Adaptive Radio (where the communication system monitors and modifies its performance) and Software Defined Radio (SDR) capable of implementing traditional communication schemes (e.g. coding, modulation, etc.) by software.

2.1.1 Spectrum Share

Radio spectrum is a portion of the electromagnetic spectrum having frequencies ranging from 300 GHz to 3 kHz. With the necessary safeguards, it can be used to improve the efficiency and productivity of the modern economies, as well as to contribute to improving the quality of citizen's life.

Given the increasing development of diversified and innovative applications, there is an increasing use of the available spectrum. It is inevitable to move towards an increasingly shared use of the radio spectrum. Cognitive Radio is the solution for the requirements of future services and the scarcity of frequencies. Services described in Figure 2.1 are currently supported by wireless communications, which require a permanent need for innovative and increasingly efficient techniques of spectrum sharing.

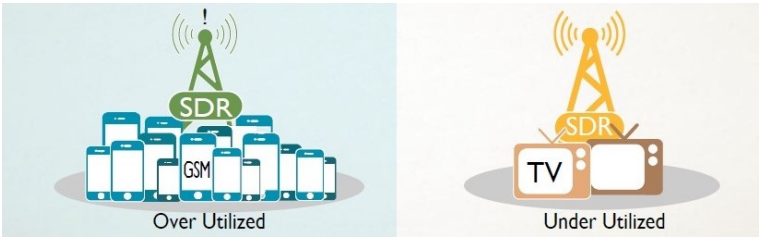


Figure 2.1: Spectrum usage. Adapted from [**Wipro**].

One of the primary principles of spectrum management is to harmonize the use of radio frequencies across borders and, wherever possible, to extend harmonization on a worldwide scale, thus improving the efficiency of the use of this resource through the reduction of the costs of the equipment and services to be offered to consumers.

2.1.2 Deployment Scenarios

When a simple voice conversation over the phone is considered, some adaptations become clearer depending on the environment and who the user is. For instance, if the user is in a crowded and noisy environment such as a stadium, the voice's perceptibility decreases drastically. Cognitive Radio is aware of the user and adapts to satisfy his needs. This means that Cognitive Radio has strong abilities such as learning and sensing beside awareness and adaptation.

2.1.3 Type of Users

The general principles of Cognitive Radio networks are to allow unlicensed wireless devices (called Secondary Users or SUs) to access the licensed frequency bands without interfering with the certified users (called Primary Users or PUs). The overarching approach in the area of Cognitive Radio is to have SUs with the ability of dynamically detect and access spectral opportunities, that is, frequency bands that are not being accessed by PUs at any given time in a given location. In this context, SUs should be able to accurately analyze radio spectrum and adapt depending on the current use of the channel by the PUs.

2.1.4 Sensing the Channel

To increase the efficiency of the system, Cognitive Radio uses sensing to evaluate the spectrum to know if someone is using it. It is essential to avoid interference. A SU needs to carry out spectrum sensing accurately and in a very short period of time. Since additional energy consumption is required for spectrum sensing, when compared to other traditional communication systems, the increase of energy efficiency of Cognitive Radio devices becomes an important issue.

2.2 Opportunistic Spectrum Access

2.2.1 Cooperative System

One of the most significant problems with spectrum sensing it's accuracy, i.e. the capabability of detecting PUs when they effectively are active and the capability of not declaring the channel as being used when it is idle. To improve the sensing performance, cooperation among the secondary users can be utilized to collect more information. By using a cooperative sensing system, the spectrum sensing accusary can be improved because a greater number of receivers will be able to build up a more accurate picture of the transmissions in the area.

The cooperative system is also important because it allows SUs to assist PUs in the data transmission. So, it is possible to use more efficiently the time slots that are not used by PUs.

2.2.2 Preemptive Priority Queuing

As we know, the the size of the packets differ from service to service. For example in voice transmitted over packets the packet length is short to decrease the delay of the packets' flow. On the other hand, for best effort traffic, such as email, longer packages are adopted because they are not delay sensitive. Figure 2.2 presents an example of a cognitive radio network, where a primary and a secondary transmitter may send voice and data packets.

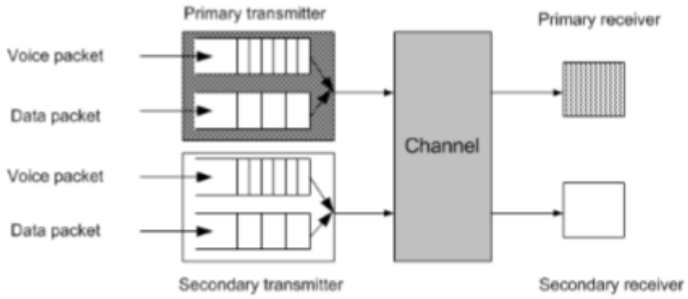


Figure 2.2: Service Packets (adapted from [3]).

Preemptive priority can offer different services based on the type of packets. In preemptive priority discipline, the service with higher priority starts sending packets on the channel, interrupting the current transmission with less priority. An interrupted packet on its re-entry may either resume from the point where it was preempted or restarts its transmission from the beginning. In addition to the already existing PU over SU, it was also considered a different priority among different services. In the case of voice packets, these have more priority than data packets due to the delay requirements.

2.2.3 Handoff

The research in Cognitive Radio networks is conceived to solve the problem of spectrum scarcity. Therefore, Cognitive Radio networks are providing particular attention to the different channels requirements. The key is to allow the dynamic use of the underutilized spectrum. Spectrum handoff can do the dynamic use of the available unutilized spectrum as it can be observed in Figure 2.3, where the transmissions in red are moved from different channels (handoff) as the time goes on.

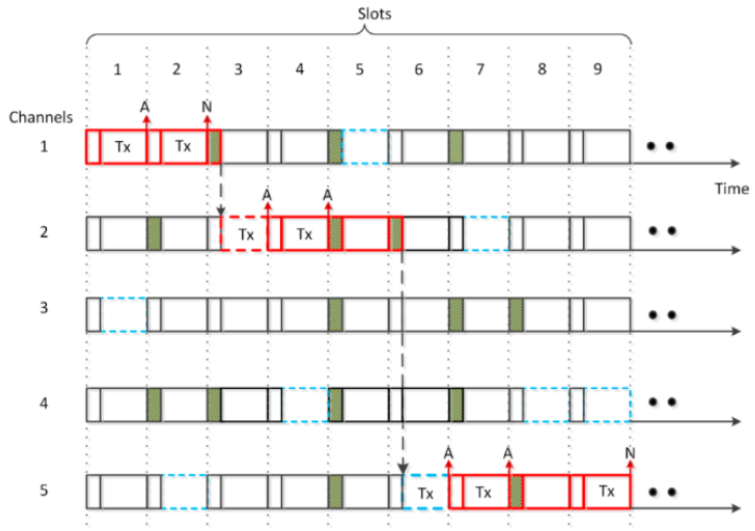


Figure 2.3: Handoff of channel (adapted from [8]).

2.3 Literature Review

In this section, we present a brief description of studies already done, including experimental results. The works report different and relevant techniques and concepts for the improvement of Cognitive Radio networks.

2.3.1 Analysis of Opportunistic Access

The work in [2] studies the waiting time and queue dynamics of Cognitive Radio networks. The waiting time is based on a model of channel time slots in which there are different slots spaced with the same time interval as presented in Figure 2.4. The model propose in the paper uses the different priorities of the distinct users to decide who transmits in these slots, so users with lower priority (SU's) will have to check if the users are using the slots with more priority (PU's).

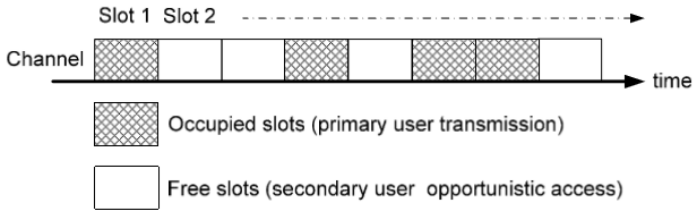


Figure 2.4: Channel time slots (adapted from [2]).

The tests performed in [2] are based on simulations under ideal conditions (ideal channel, i.e. communication errors do not occur), and it was assumed that each packet would occupy only one channel time slot. Transmission occurs at the beginning of a time slot. It is also considered that the buffers are of infinite size (waiting list). Two simulations were made. In a first scenario the rate of packet transmissions adopted by the two users is equal. In the second simulation the primary user adopted a fixed transmission rate while the second one adopted a variable rate.

Figure 2.5 depicts two curves: one red for the PU and one black for the SU. We see that the average waiting time of the SU increases while the waiting time of the PU remains constant. Moreover, the waiting time of the SU increases exponentially with the packet arrival rate. As expected in this first scenario, as the primary user arrival rate increases, so does the average waiting time for the secondary user.

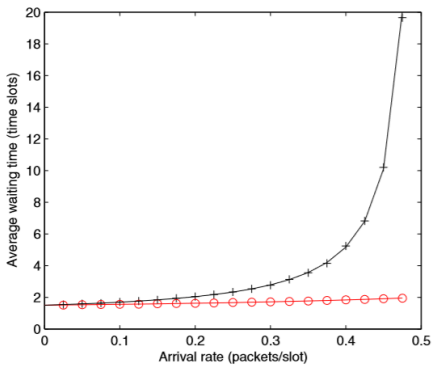


Figure 2.5: Average waiting time for the primary and the secondary users. Solid lines are theoretical results and markers denote simulation results (adapted from [2]).

We can observe in Figure 2.6 that there are diferent colors for diferent arrival rates of packets for the PU (λ_P). The increase of λ_P and SU arrival rates of packets (represented in the X axis) influences the average waiting time of the secondary user. It is also possible to note that the probability methods used in this paper are positively accurate.

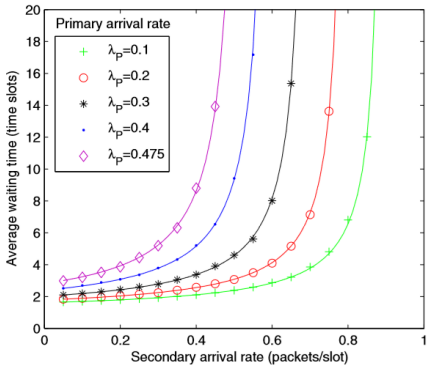


Figure 2.6: Average waiting time for the primary and the secondary users. Secondary user arrival rate is varied for different fixed arrival rate for the primary user. Solid lines are theoretical results and markers denote simulation results (adapted from [2]).

[2] also characterizes the impact of the imperfect sensing has in Cognitive Radio networks. The same is studied in [6, 9]. Imperfect sensing means the possibility of not guarantying 100% of probability of detection (PD) and 0% of probability of false alarm (PFA). Figures 2.7 and 2.8 show the primary user and secondary user's packet delay. By assuming imperfect sensing, it is considered that a SU may detect a free slot as occupied (false alarm) and a second case where it may recognize an occupied slot as being idle(mis-detection), interfering with the transmission of the PU. Both situations increase the delay to the users.

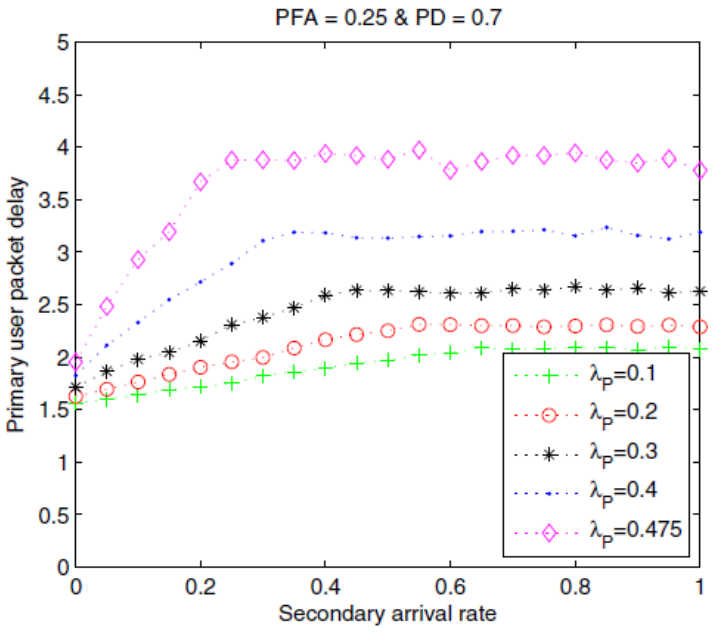


Figure 2.7: Primary user packet delay (adapted from [2]).

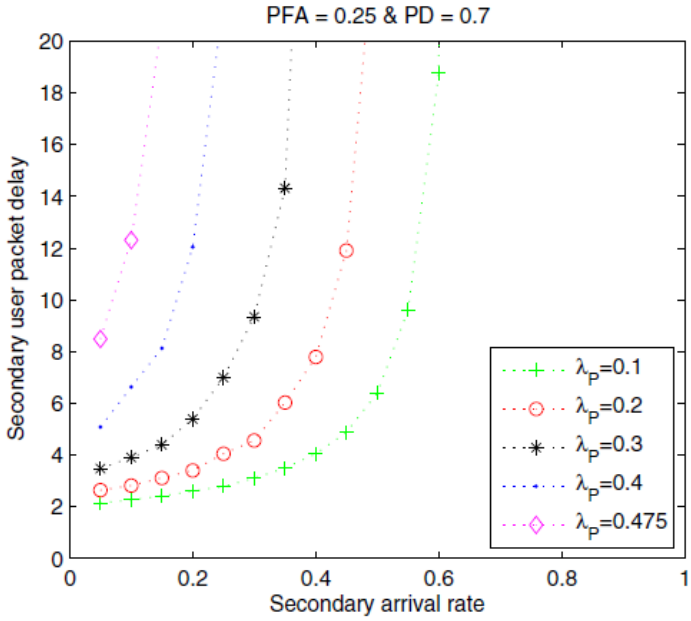


Figure 2.8: Secondary user packet delay (adapted from [2]).

Comparing Figure 2.8 with Figure 2.6 it is possible to see that imperfect sensing increases the packet delay for secondary user.

2.3.2 Cooperative with Priority Queueing Analysis

The report [1] analyzed the hierarchical structure used in Cognitive Radio networks with the cooperation between the users. It also studied the impact of cooperation between secondary and primary users. The cooperative scenario is illustrated in Figure 2.9.

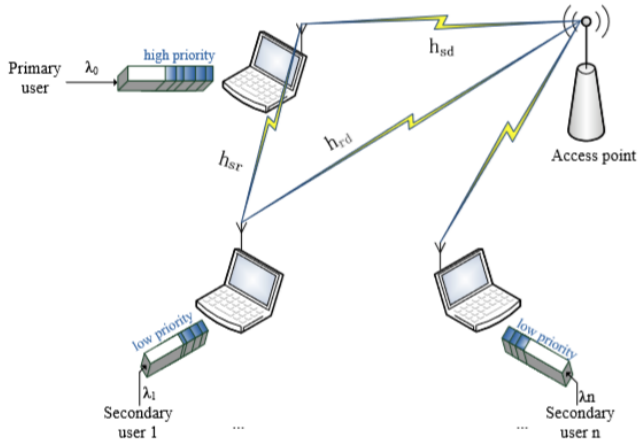


Figure 2.9: Cooperative cognitive radio networks with queueing model (adapted from [1]).

In the study a primary user (PU) and n secondary users (SU) were created. The PU always has priority in the channel. The spectrum access policy of the SUs relies on a FIFO (first-in-first-out) medium access control (i.e. the first SU to have a packet is the first one to transmit).

It is known that Cooperative Diversity allows decreasing the packet error rate since it reduces the fading problem. Two models were formulated to study the importance of cooperative systems in Cognitive Radio networks.

In the non-cooperative model based on the priority queueing system it is considered a typical case of Cognitive Radio, that is, we have two users (primary and secondary). Both respect their status, meaning that whenever the primary user is sending packets, the secondary is sensing the channel to check if it is vacant. The simulation is performed assuming an M/G/1 queueing theory model, i.e. a queue where the input process (arrivals) is memoryless/Markovian (M) process and the service time follows a general (G) distribution..

The model with Cooperative Diversity tries to face the fading effects (including path loss) that can increase the number of packages not successfully received. In this scenario it is possible to quantify the gains obtained by cooperation between users (primary and secondary). The authors considered the scenario of having only one PU, one SU and one AP (Access Point).

The PU sends his data to different SUs to increase the level of diversity and the SUs retransmit the PU packets to the sink. Using this model, the primary user dispatches the packets to the queue quicker so that the SUs can access the channel faster.

Initially, the two models were evaluated considering a high quality transmission channel. Figure 2.10 shows the throughput obtained with relaying and without relaying (no cooperation).

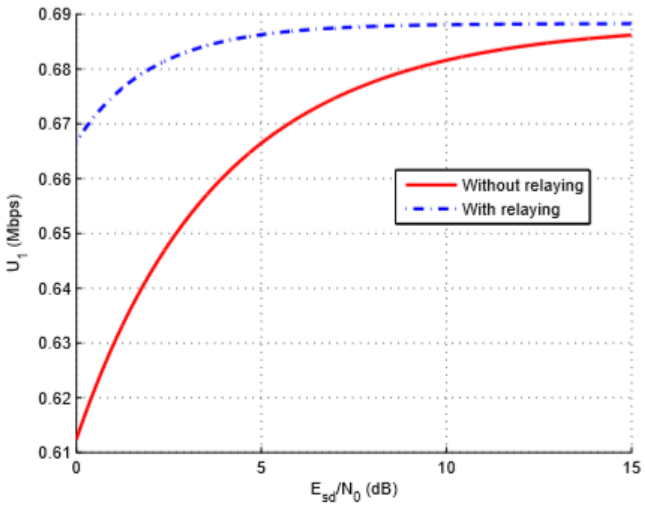


Figure 2.10: Throughput of the secondary user with a rate of 30 Packets/Sec (adapted from [1]).

Observing the differences between the two throughputs of the secondary user, we conclude that cooperation always drive higher throughput values and the performance gain is higher for lower SNR conditions. Figure 2.11 shows the improvement of the secondary throughput with different values of packet traffic when the quality of the transmission channel between the primary user and the AP is significantly decreased.

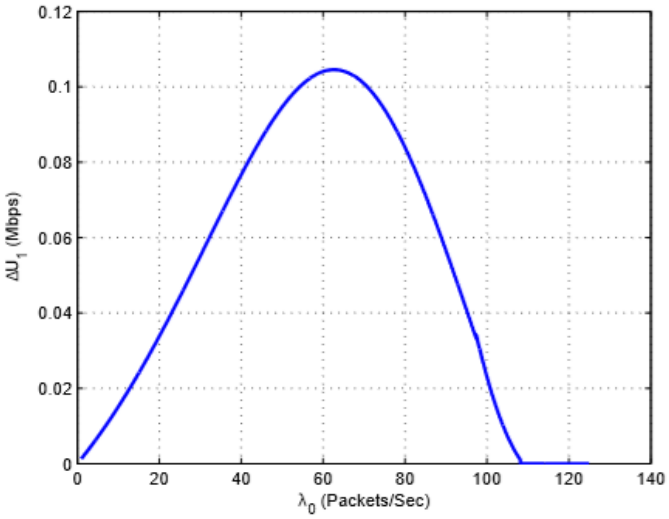


Figure 2.11: Throughput gains of the secondary user with respect of the primary traffic loads (adapted from [1]).

As observed in Figure 2.11, the throughput gain varies according to the traffic generated by the primary user (λ_0). When we have low λ_0 the SU has the channel with more idle time slots, not needing relaying. When we have high λ_0 values the spectrum opportunity is scarce therefore the gain in relaying is low. The maximum throughput gain is achieved when the average of generated traffic is close to 60 packets per second.

2.3.3 Average Waiting Time of Packets with Different Priorities

The work in [3] studies the average waiting time using a preemptive priority queuing system. The system considered in [3] is the one described previously in subsection 2.2.2. Two scenarios were considered: a first one in which the secondary user (SU) senses at the beginning of each slot (classic Cognitive Radio); and another in which the SU makes continuous sensing. The impact of packet sizes and packet types is analyzed in both scenarios.

Endless waiting lists and FIFO (first-in-first-out) processing as considered. In addition to the existing PU priority over the SU, it was also considered a service priority where voice packets have a higher priority than message packets because of the required quality of service (latency).

In the first scenario, we have multiple PUs and only one SU. We are in the typical situation in which the different PUs start sending at the beginning of each time slot.

Whenever the SU detects that the channel is free, it starts sending its packets. It is assumed that the SU makes perfect sensing and the sensing time is small (almost negligible) as it is represented in Figure 2.12.

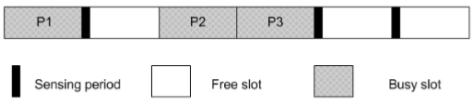


Figure 2.12: TDMA with equidistant time slot (adapted from [3]).

In the second scenario a single PU and a single SU is considered and an example of channel access is depicted in Figure 2.13. The SU performs continuous sensing and starts transmitting whenever the channel is sensed idle. It does not prevent the PU from entering the channel because when the PU accesses, the SU will interrupt its transmission immediately.

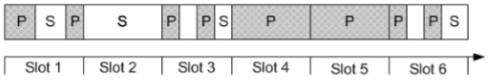


Figure 2.13: Example of channel access for single PU and single SU (adapted from [3]).

The average packet waiting time was studied for voice and data packets, considering the scenario 1. The results are illustrated in Figure 2.14. We can observe that increasing the PU's arrival rate of packets per slot increases exponentially the average waiting time of the SU due to the high number of time slots that become occupied. Moreover, as the PU's arrival rate increases the data packets suffer higher average waiting time than voice packets.

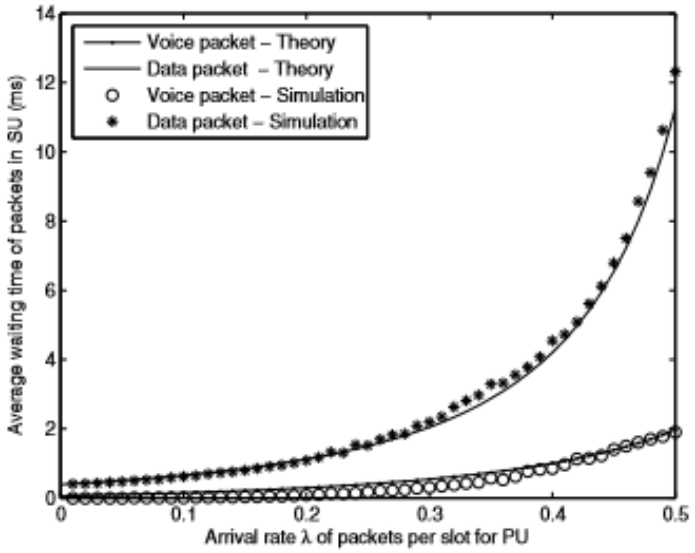


Figure 2.14: Average waiting time of packets in SU (the SU's arrival rate of voice packets per slot is 5; the SU's arrival rate of data packets per slot is 2) (Adapted from [3]).

In the second scenario, they observed that the waiting time of the PU voice packets is not significantly affected. This fact is due to voice packets, which have higher priority over the data packets, being transmitted without experiencing a longer waiting time. As for data packets, a small increase in the waiting time is also noticeable due to the proposed packet priority system. We can also observe in Figure 2.15 that the average waiting time of the SU is different when compared with the first scenario.

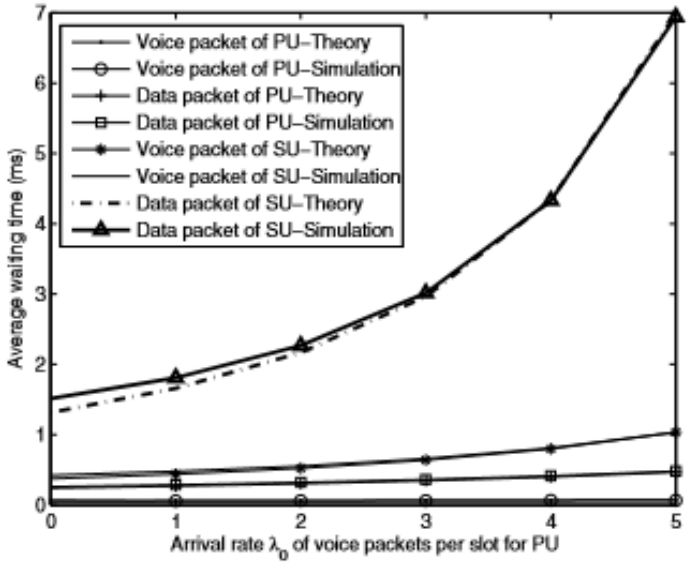


Figure 2.15: Average waiting time of packets of both users type in a scenario with single PU and single SU using four different packet priority classes (adapted from [3]).

Because of the small size of voice packets and large system bandwidth, some of voice

packets can be transmitted in a short time. As such, the queue of voice packets may require only a small portion of a time slot to return to be empty. The remaining part of a time slot can then be utilized by packets with lower priority (data packets of PU, voice packets of SU, data packets of SU). This leads to a reduction of average waiting time of the lower priority services. Hence, scenario two offers a better spectrum utilization compared to scenario one.

2.3.4 Optimal Access Strategy in Multichannel Dynamic Spectrum

Work [5] focuses on the delay performance analysis of a secondary user (SU) and the best strategies for accessing multiple dedicated channels. The optimal access advocated in this study is modeled as a nonlinear programming process that can be solved using standard Genetic Algorithms functions.

2.3.4.1 Simulation Model

The model studied in this paper is a typical hierarchical system of Cognitive Radio in which there are two types of users. Primary ones that always have priority and the secondary ones only transmitting in when PUs are absent.

It is assumed that there are M parallel licensed channels, N SUs with ids from 1 to N, and each PU only transmits on its dedicated channel. The SUs can communicate on any channel and also do perfect sensing as considered in Figure 2.16. The SUs immediately defer their transmission when the PU starts to use the channel. This model also considers Centralized Scheduling: the traffic between SU's are scheduled so that there are no collisions between them. An M/G/1 queueing model is adopted.

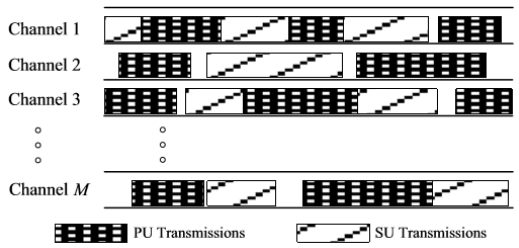


Figure 2.16: Channel occupation of PU and SU transmissions (adapted from [5]).

To simplify the analysis, when a SU has a packet to transmit it is assumed that the packet can only be transmitted in a single channel. In other words, a SU do handoff to another channel only when a packet transmission ends.

Figure 2.17 shows the SU traffic system delay for different values of SU and PU traffic density. It can be seen that the increase of SU traffic system delay is approximately exponential, meaning that the system delay increases with the SU traffic density.

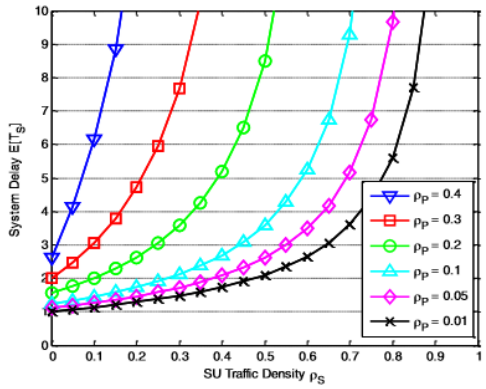


Figure 2.17: Trend of SU system's delay under different SU and PU traffic density (adapted from [5]).

Observing Figure 2.17 we can create a relation between the densities of the primary and secondary traffic with the delay. The higher the density will be the higher delay of SUs.

In the scenario where a SU has to choose between two different PU channels, the SU must use a metric that allows it to have the lowest average delay, thus creating a probability-based access rule.

For the purpose of illustration, it was considered the case of SU traffic accessing two licensed channels. When equal or non-equal PU traffic arrival rates are set in the two licensed channels, Figure 2.18 and Figure 2.19 show the corresponding optimal access probability, validating the conclusion that an optimal access strategy is possible.

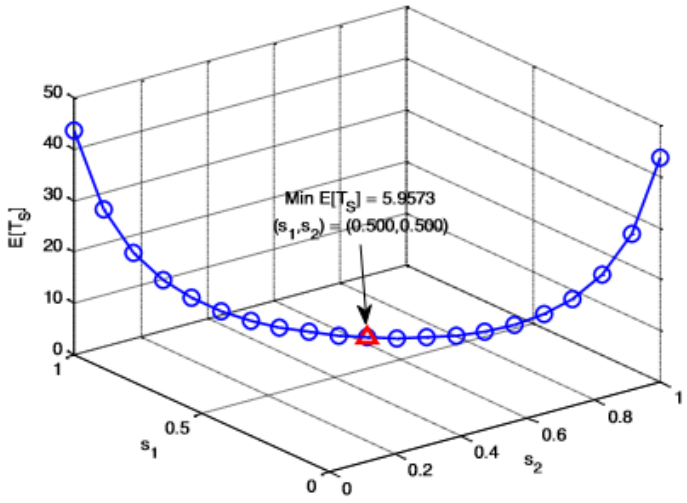


Figure 2.18: Expected system delay when PU's traffic arrival rate is the same (adapted from [5]).

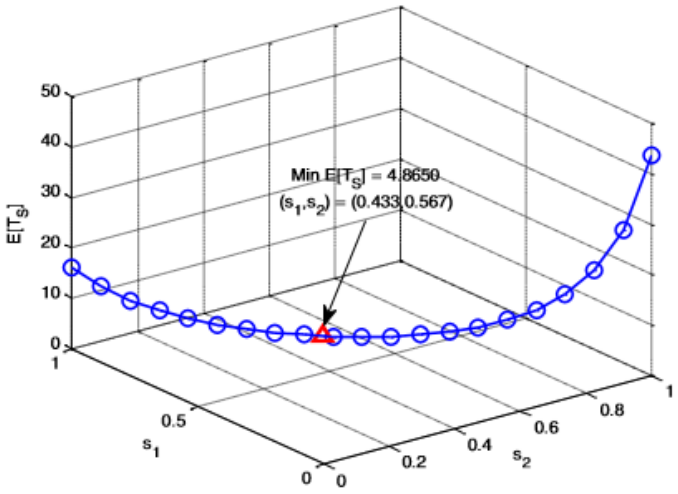


Figure 2.19: Expected system delay when PU's traffic arrival rate is different (adapted from [5]).

As we can see in Figures 2.18 and 2.19, the minimum waiting time is achieved for probabilities that make the SUs access more often to the channel and with less traffic, thus avoiding excessive traffic in a single channel, obtaining a reduced waiting time.

Finally, the authors have simulated a SU accessing a different number of channels to gather different values of system delay, as is represented in Figure 2.20.

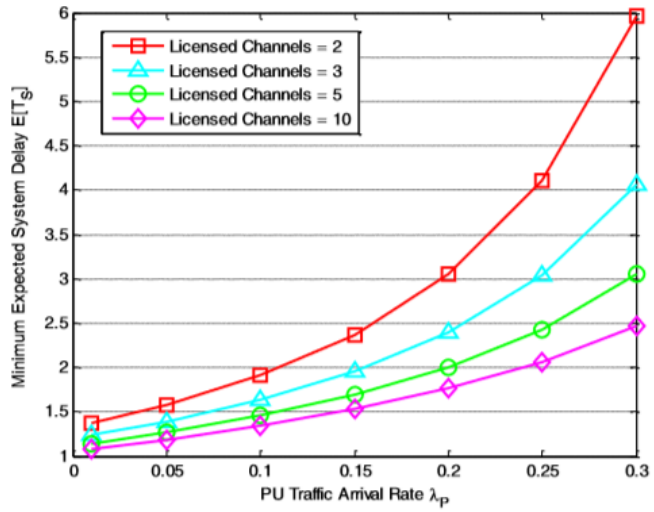


Figure 2.20: Comparison of the minimum expected system delay for the case with different licensed channels when PU traffic arrival rate is the same (adapted from [5]).

In Figure 2.20, we can observe the difference that a handoff system makes in a cognitive radio network. Thus, we can say that if an SU has the opportunity to access many different channels, it will decrease its waiting time.

This work still needs to study the scenario of handoff with multiple SU's to verify if the collision of packets transmitted by them will damage this system of probabilities.

2.3.5 Characterization of the Opportunistic Service

Work [10] performs a study of the service time that a SU requires to transmit a packet using the licensed radio spectrum of a primary user. In a Cognitive Radio network, it is possible that during a SU's transmission multiple interruptions caused by the PUs may occur. The aim of this work is to derive the characteristic function and the distribution of the service time. This can be done by deriving the probability of a SU transmitting its packet when k $>$ 0 periods of PU's inactive states.

The model used for this study is a wireless channel with one pair of PUs (sender and receiver) and one pair of SUs (sender and receiver) trying to access the channel in a opportunistic way. The PU transmitter has two states: ON when it is active (i.e., transmitting) during μ_B and OFF when it is inactive (not using the channel) during μ_I. This channel usage is described in Figure 2.21.

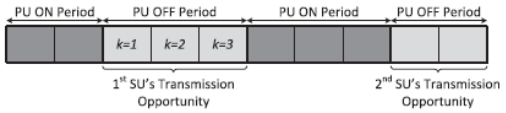


Figure 2.21: Hypothetical sequence of PU OFF and ON periods (adapted from [10]).

It is also studied a scenario where we exists multiple SUs in the same wireless channel. It was considered an ideal MAC scheme where the SUs with transmition queues are scheduled to access the channel instantaneously (i.e., the MAC coordination time is null). This model is described in Figure 2.22.

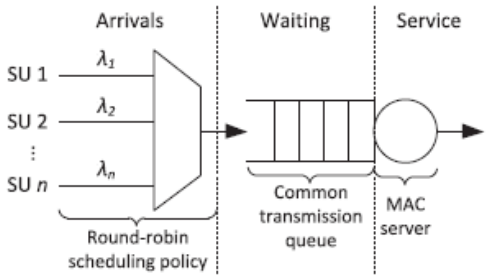
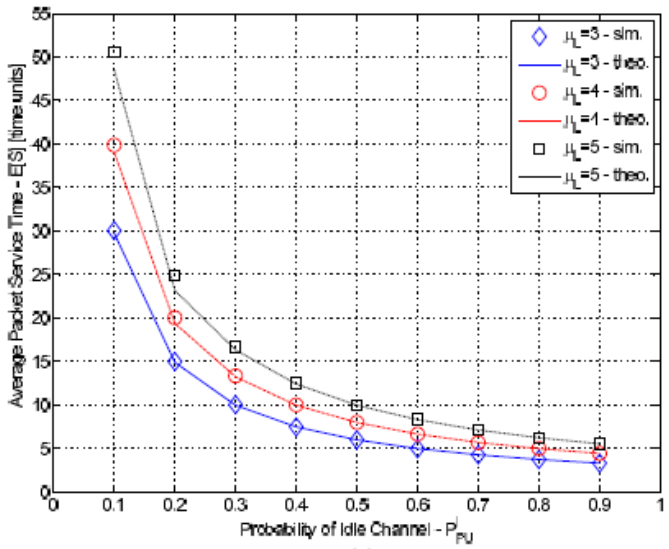


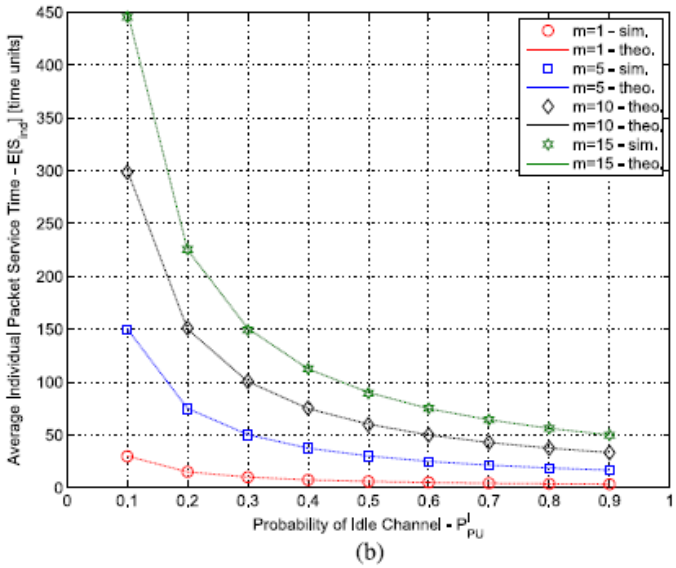
Figure 2.22: Conceptual model to derive the packet service time when multiple SUs are cheduled for transmission (adapted from [10]).

Figures 2.23 and 2.24 show the average packet service time for both single and multiple (m) SUs scenarios under saturated condition. Different SU's packets durations (μ_L) and PU's inactivity rates (P_{PU}^I) are considered.



(a)

Figure 2.23: Average packet service time of a single SU for different PU's inactivity rates. The average PU's frame length is set to $\mu_B + \mu_I = 10$ time units and for the multiple SU scenario the SU's mean packet duration is set to $\mu_L = 3$ time units (adapted from [10]).



(b)

Figure 2.24: Average packet service time of multiple SUs for different PU's inactivity rates. The average PU's frame length is set to $\mu_B + \mu_I = 10$ time units and for the multiple SU scenario the SU's mean packet duration is set to $\mu_L = 3$ time units (adapted from [10]).

The results show that the theorical model derived in this study is successfully validated by the simulations results. It can be also observed that when the number of SUs increases, the average packet service time also increase, as a consequence of the adopted MAC policy.

CHAPTER 3

Service Time Model

3.1 System Model

One of the objectives of this work is to use a mathematical model that characterizes the service time of a Cognitive Radio channel.

The model starts with the derivation of an expression for the Probability Generating Function (PGF) of the packet service time. The packet service time, measured in discrete units of time, is represented by the random variable (r.v.) S, while its PGF is denoted by $G_s(s)$. It is assumed that a packet takes one unit of time to be transmitted. Moreover we assume that the channel is consecutively idle during μ_I units of time and consecutely observed as being occupied during μ_B consecutive units of time

The definition of service time of a SU's data packet is the interval from the instant when a packet arrives the head of the transmitter's queue, until the instant when its transmission ends. If the packet arrives at the head of the transmitter's queue during an idle period it will be transmitted without any delay, and the packet service time is given by the length of the data packet. However, if the packet arrives at the head of the transmitter's queue during a busy period, the service time will have to account with the remaining busy period duration. Therefore, the PGF of the service time is given by

$$G_s(s) = \gamma_I G_L(s) + (1-\gamma_I) G_L(s) G_B(s), \tag{3.1}$$

where G_L represents the PGF of the lenght of the data packet while G_B represents the PGF of duration of the busy periods. γ_I represents the probability of a packet arriving at the head of the transmitter's buffer queue during an idle period. Under saturated network conditions, a new packet arrives at the head of the SU's buffer queue during an idle period in the last $\mu_I - 1$ units of time that lasts the idle period. This is because a node has always a packet to transmit (saturated traffic) and the packet served during the first unit of time

of the idle period suffers a longer delay during the busy period (the period of time during which the channel is busy). γ_I is given by

$$\gamma_I = \frac{\mu_I - 1}{\mu_I}, \tag{3.2}$$

where μ_I represents the average time (in time units) the channel is consecultively found idle. Under non-saturated network conditions, a new data packet may arrive randomly at any time unit of the channels idle period, and therefore γ_I is given by

$$\gamma_I = \frac{\mu_I}{\mu_I + \mu_B}. \tag{3.3}$$

Using the total law of probability we can write the probability of a packet arriving at the head of the SU transmitter's queue during an idle period for both saturated and non-saturated network conditions as follows

$$\gamma_I = (1 - p_{QE})\frac{\mu_I - 1}{\mu_I} + p_{QE}\frac{\mu_I}{\mu_I + \mu_B}, \tag{3.4}$$

where the left-hand side represents the probability of a SU having a packet to transmit in a traffic saturation condition and the right-hand side represents the probability of a SU having a packet to transmit when it is not saturated. After some mathematical simplifications we can rewrite γ_I as

$$\gamma_I = 1 - p_I + p_{QE}(-1 + p_I + P_{PU}^I). \tag{3.5}$$

As stated in Section II, in this work we assume that SU's data packets have a fixed length of 1 time unit, meaning that the PGF of the packet length, G_L, is simply given by $G_L(s) = s$. On the other hand, because it is assumed that both busy and idle period durations are distributed according to geometric distributions, the PGF of the busy period duration is given by

$$G_B(s) = \frac{p_B s}{1 - (1 - p_B)s}. \tag{3.6}$$

Using (4.5) and (4.6) in (4.1) we can rewrite the PGF of the packet service time as follows

$$G_S(s) = (1 - p_I + p_{QE}(-1 + p_I + P_{PU}^I))s + (p_I - p_{QE}(-1 + p_I + P_{PU}^I))\frac{p_B s^2}{1 - (1 - p_B)s}. \tag{3.7}$$

The PGF of the service time presented in (3.7) includes the probability of a SU not having a packet to transmit p_{QE}. However, under non-saturated network conditions, the value of p_{QE} depends on the packet arrival rate of each SU and on the time needed to transmit each packet, i.e., the average packet service time. Therefore, to define p_{QE} we

adopt a queueing model, which is responsible for the transmition queue statistics. A queueing model M/G/1 can be considered if using a Poisson distribution with an average arrival rate λ (packets/time unit), for the assumption of the packet arrival process of the SU. Also a single data channel is used for transmission (number of servers = 1). The PGF of the queue length of a M/G/1 queueing model is given by

$$G_Q(s) = \frac{(1-s)(1-\rho)\mathscr{L}_S(\lambda(1-s))}{\mathscr{L}_S(\lambda(1-s)) - s}, \tag{3.8}$$

where $\mathscr{L}_S$ is the Laplace transformation of the service time distribution. The replacement of s in (3.8) by e^s results in $\mathscr{L}_S$. The variable ρ expresses the traffic intensity,

$$\rho = \lambda E[S], \tag{3.9}$$

and $E[S]$ represents the mean service time given by

$$E[S] = G'_S(s)\big|_{s=1}. \tag{3.10}$$

The probability of finding the queue empty is given by

$$p_{QE} = \frac{G_Q(s)^{(0)}}{0!}\Big|_{s=0}, \tag{3.11}$$

as well as the expressions in (3.10) and (3.9), to define the following system of non-linear equations

$$\begin{cases} \rho = \lambda E[S] \\ p_{QE} = \frac{Q(s)^{(0)}}{0!}\big|_{s=0} \\ E[S] = G'_S(s)\big|_{s=1} \end{cases}, \tag{3.12}$$

where p_{QE}, ρ and $E[S]$ are unknown. For a given λ we can solve the system numerically and compute the probability of finding the queue empty p_{QE} as well as the mean packet service time $E[S]$.

Finally, the distribution of the service time can be obtained by taking the derivatives of the PGF of service time, G_S, as follows

$$Pr\{S = k\} = \frac{G_S(s)^{(k)}}{k!}\Big|_{s=0} \tag{3.13}$$

Consequently, the resulting Probability Mass Function (PMF) of the packet service time is:

$$Pr\{S=k\} = \begin{cases} \dfrac{p_B + p_I(-1+p_I+p_B)(-1+p_{QE})}{p_I+p_B}, & k=1 \\ \dfrac{(-1)^{k-1}(1-p_B)^{k-2}(p_I p_B(-p_{QE}+p_I(-1+p_{QE})+p_B(-1+p_{QE})))}{p_I+p_B}, & k>1 \end{cases} \tag{3.14}$$

3.2 Validation and Numerical Results

This section presents a set of simulations and numerical results to validate the mathematical model described in section 3.1. Considering a M/G/1 queueing model, the packet's average service time for different λ values is also validated. After the setup of the auxiliary channels, we simulated the service time for different arrival rate of SU packets per slot.

3.2.1 Assumptions

In order to carry out these simulations, some variables were assumed in order to simplify this process. Packages issued by the PU and SU have a fixed size of 1 slot. The transmission averages are performed every 20 packets, that is, if we say that the PU transmits, on average, 20%, means that in 20 continuous time slots, on average, the PU uses 4 to transmit packets. The channel where the interactions between the PU and the SU are being simulated has a size of 10^6 time slots. Each service time result is the average of 3 simulations. Then there are 3×10^6 interactions for each different value of packet transmission by the SU and the PU. We started the simulation by creating multiple channels with different levels of occupancy by the primary user. These levels were changed from 10% to 90% of occupancy. Using a 10% level, we have a mean number of idle slots equal to 18 slots and a mean number of busy slots equal to 2 slots. Another situation assumed is that the SU performs a perfect sensing to the channel, that is, it always knows if the PU will send or not packets in the channel. Finally, it was assumed that the queue has infinite size, that is, the packets are not discarded.

3.2.2 Simulations

The simulations were performed using MATLAB, and the script used in the simulations is in the Annex I. This script is divided in three parts: - Simulation of a channel only used by PU; - Simulation of a channel only used by SU; - Simulation of a channel used by PU and SU.

The simulations start by creating an auxiliary channel where a single PU may use it to transmit. If the PU has, for instance, a transmission rate of 50%, this channel will have 50% bits with value 1, as well as, for 0. The bits with value 1 results in time slots when

the channel is busy sending a package to the receptor; whereas, bits for 0 represents time slots when the channel is free for sending packages (the packet has a time slot size of 1, discrete time).

After this, it was created in other similar cycle another auxiliary channel for SU. In this channel, the PU was not able to send any package at all. For example, if the SU had a 75% transmission rate, the channel would be 75% occupied.

Finally, it has been done an algorithm in order simulate a channel where PU and SU can transmit, forming a channel with the results of the transmissions from both users. Figure 3.1 represents a diagram with the flow of the algorithm.

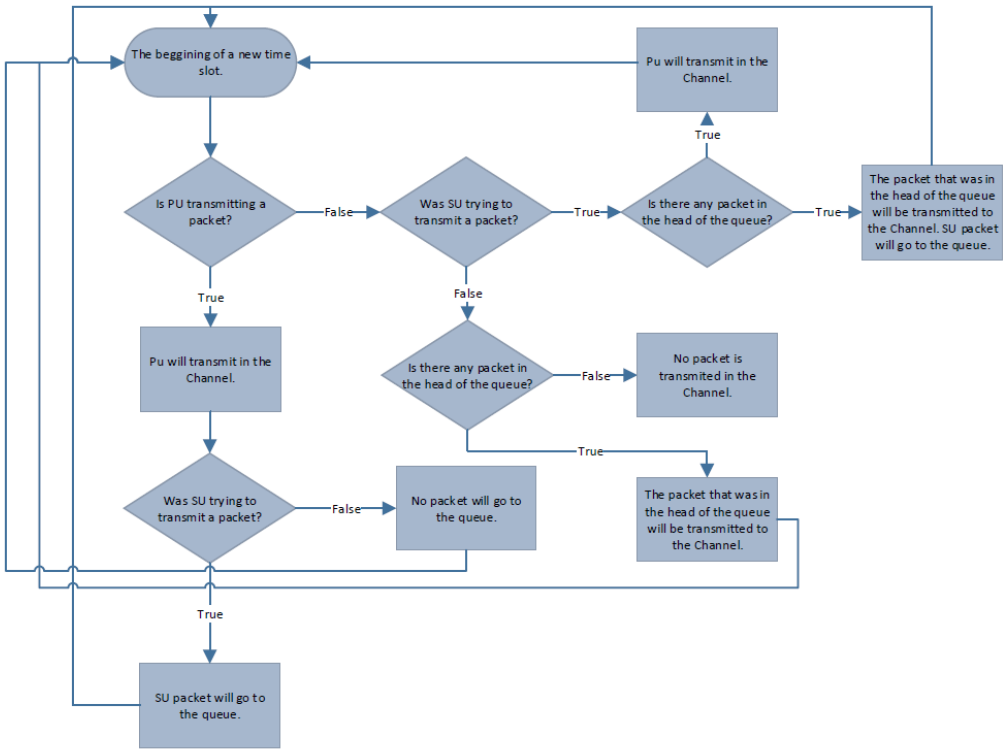


Figure 3.1: Simulation Algorithm.

3.2.3 Evaluation Results

Figure 3.2 compares the theoretical average packet service time (numerical results - curves) with the simulation results (markers) for a single SU scenario.

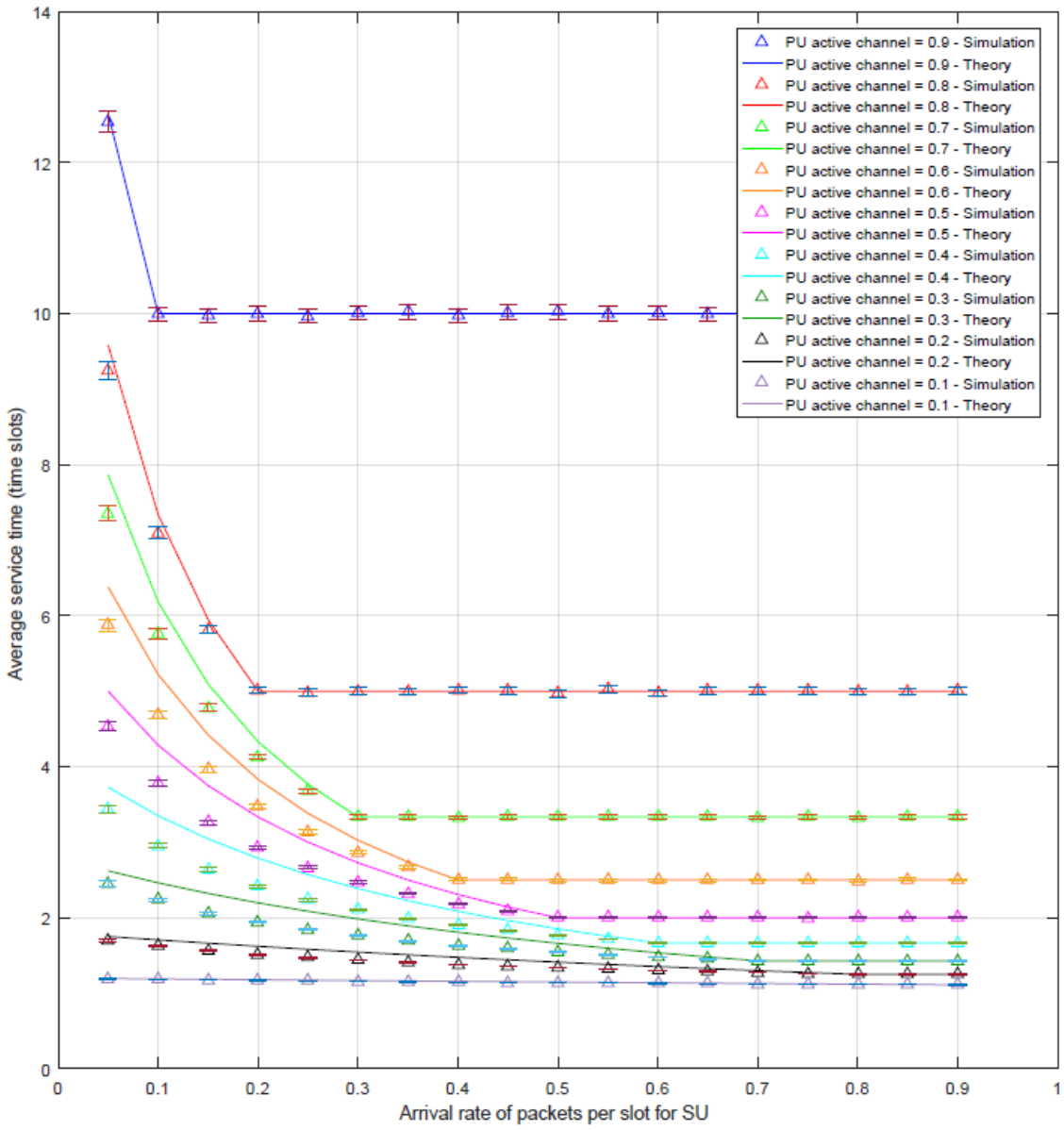


Figure 3.2: Average packet service time for a single SU scenario for different PU's activity rates.

The results obtained show that the simulation results are close to the numerical results. However, between 20% and 60% of arrival rate the SU's average service time obtained through the theoretical model is slightly higher when compared to the simulations results. It is also shown that when the arrival rate of packets per slot for SU increases, the average packet service time decreases, as a consequence of gathering more samples as the arrival rate of packets increase. We also highlight the behaviour of the service time when the queue is always busy. In this scenario, the service time remains constant, since the channel is continuously being used to transmit packets.

CHAPTER 4

Channel Selection Algorithms

Cognitive Radio networks must have the ability to learn and adapt their wireless transmission according to the ambient where they are operating. Algorithms for channel selection are therefore essential for the implementation of this technology. Generally, these algorithms analyse the environment around them and store some variables to build knowledge about the environment. After gathering some data, these Algorithms start to learn and adapt their decisions to use the spectrum as receivers or transmitters.

After learning, the cognitive device is required to make decisions involving actions that will enable to adapt itself to its surrounding environment (and sometimes modify this environment). This decision comprises the choice of the action to be performed. The choice will be guided by the information acquired by the system and, in particular, by the information about other network devices. It is particularly essential that each device knows or owns a priori knowledge about the strategy adopted by other network devices, to avoid reaching a never-ending situation in which the device decides to take this or that decision because other devices are not working as assumed [4].

This chapter introduces the problems associated to the decision making and learning in the context of CR. In particular, the concept of a cognitive agent will be introduced. Then, the constraints related to decision making introduce the notion of decision space. Subsequently, decision making and learning will be discussed from the device and network viewpoint.

4.1 Sensing

There are several different ways to sense the occupation of PU's in a channel. One of the methods is called Energy Detection (ED) [11]. ED is a method to detect a signal through the ammount of received energy. H_1 (occupied channel) describes the case when signal plus noise and H_0 (unoccupied channel) describes the case when only noise is observed on the channel, meaning that the channel is idle.

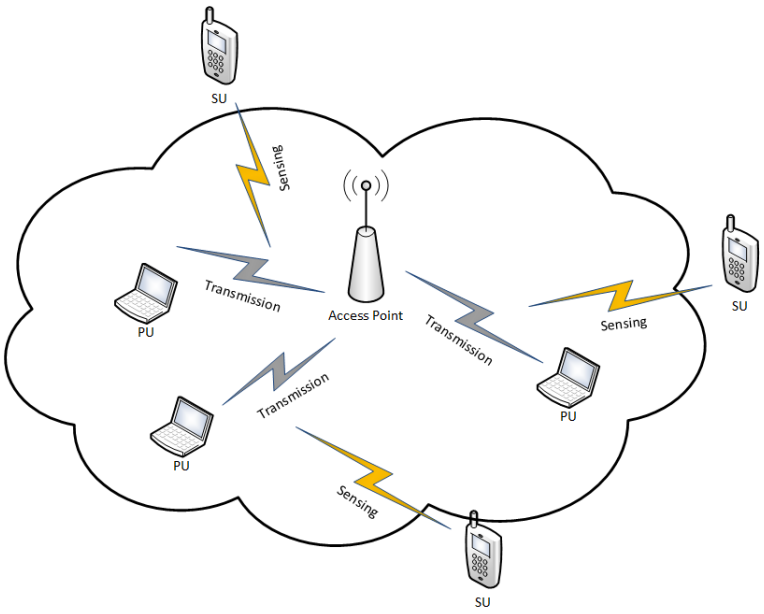


Figure 4.1: Sensing using detection of ED.

The first thing to use ED is to determine the amount of received energy using the equation:

$$E = \sum_{n=1}^{N} | x(n) |^2 \tag{4.1}$$

where E is the statistical tests used in ED and N is the number of samples. In ED sensing method, the fact that the detector has to have knowledge about the noise power, can be seen as a weakness.

CRs may declare an unoccupied channel while at the same time a neighbor / receiver declares it occupied and can successfully demodulate the signal. This problem may be because of inadequate detection sensitivity on the part of the SU, or it may be due to the hidden node problem, in which the signal is severely attenuate during its propagation till reaching the SU. Figure 4.2 illustrates the hidden-node problem. In this figure, the SU is attempting to detect all the primary user's signal. It can successfully detect PU 2 transmission but fails to detect PU 1. The sensing is characterized by strong fading or shadowing due to different propagation effects.

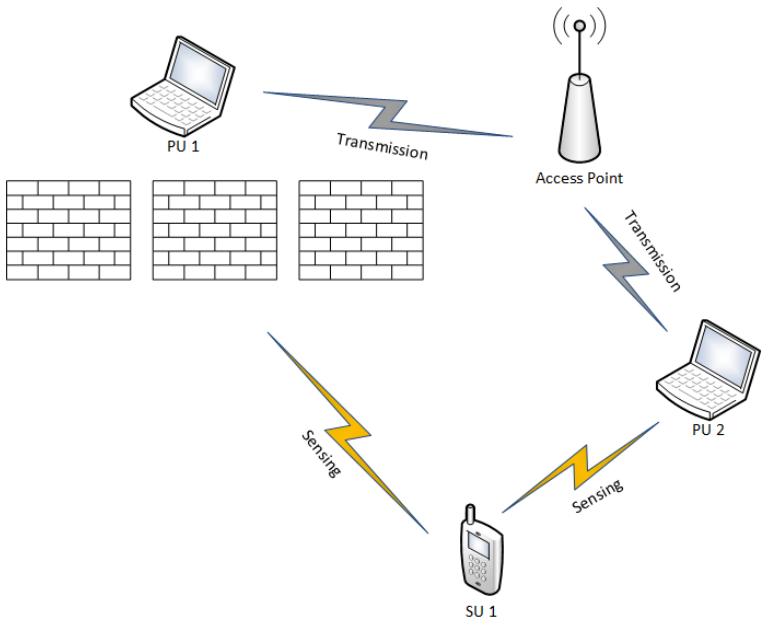


Figure 4.2: Sensing failed.

The ED process starts with cognitive users sensing the same primary user's frequency band. The channel model of cognitive users can be described as follows.

$$x(t) = \theta h s(t) + u(t), \tag{4.2}$$

where $s(t)$ is the signal sent by the primary user, $x(t)$ is the signal received by a cognitive user from the primary user, and $u(t)$ is the additional Gaussian white noise. h is the channel gain, and θ is the indicator of a primary user. When we get $\theta = 1$ it means that we are in the presence of a PU and $\theta = 0$ means a PU absence. When the primary user is absent, there is only noise floor. When the primary user is active, there is the primary user's modulated signal added with noise floor. The detection is a binary signal detection problem, which can be modeled with the following hypothesis testing for a cognitive user:

$$H_0 : x_i(t) = u_i(t) \tag{4.3}$$

$$H_1 : x_i(t) = h_i s(t) + u_i(t), \tag{4.4}$$

where H_0 and H_1 indicate the absence and presence of the primary user, respectively.

We can measure the correlation between the vision of a channel by the different nodes to evaluate this situation.

4.1.1 Measuring Correlation

According to [7] correlation is the degree to which two quantities are linearly associated. A correlation of 1 means that the linear relationship is perfect, while a correlation of 0 typically indicates independence. To evaluate the impact of the different variables of this

system it is used a metric in order to characterise the dissimilarity accomplished by the different SUs at the same instant of time.

To compute the correlation between channels (n >= 2 channels) it is created a correlation matrix of the form:

$$\begin{bmatrix} 1 & x_{1,2} & x_{1,3} & \dots & x_{1,n} \\ x_{2,1} & 1 & x_{2,3} & \dots & x_{2,n} \\ x_{3,1} & x_{3,2} & 1 & \dots & x_{3,n} \\ & & \dots\dots\dots\dots & & \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & 1 \end{bmatrix}, \tag{4.5}$$

where x_{ij} the channel decision (idle/busy) is the sample Pearson's correlation coefficient between SUs i and j. Note that this matrix is symmetric because $x_{ij} = x_{ji}$. The sample Pearson's correlation coefficient between SU i and j is given by

$$x_{ij} = \frac{\sum_{k=1}^{l} (\overrightarrow{P}_{I,i}(k) - \mathbb{E}[\overrightarrow{P}_{I,i}])(\overrightarrow{P}_{I,t}(k) - \mathbb{E}[\overrightarrow{P}_{I,t}])}{\sqrt{\sum_{k=1}^{l} (\overrightarrow{P}_{I,i}(k) - \mathbb{E}[\overrightarrow{P}_{I,i}])^2} \sqrt{\sum_{k=1}^{l} (\overrightarrow{P}_{I,t}(k) - \mathbb{E}[\overrightarrow{P}_{I,t}])^2}} \tag{4.6}$$

where $P_{I,i}$ represents a vector with consecutive l channel decisions achieved by a SU i in a finite period of time.

4.2 System Assumptions

The system assumed to demonstrate the proposed algorithms is based on three channels with a primary user operating on each channel. There are three secondary users with different values of packet generation rate (λ). The goal of the three secondary users is to transmit information over one of the three channels that can be found idle. There is also a scheduler responsible for deciding and selecting for each channel a different SU as it is shown in Figure 4.3.

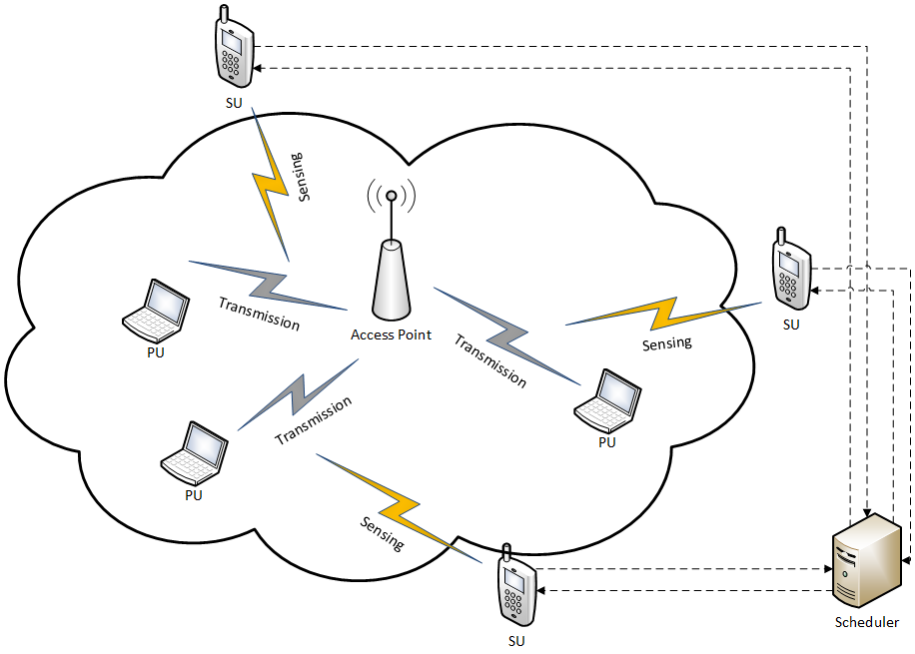


Figure 4.3: Cognitive radio scenario, where a scheduler is capable of allocating the SUs to the available channels.

The packet duration of either the PUs or SUs is a unit of time. The sensing done by the SUs is perfect, and on each channel there is only one PU, and only one SU can use the channel at the same time. The average occupation of the first channel is 18 slots in 20. The occupation rate of the second channel is 14 slots in 20. The third channel is occupied 6 slots in 20. Nodes can have an estimate of the service time for each of the channels by applying the model presented in Chapter 3. The packet generation rate (λ) of each node is known. Four algorithms will be presented for four different scenarios.

4.3 First scenario

In the first scenario, it is proposed an algorithm that, computes the average values of occupation and observes the rate of packet generation at a given instant. Then the different SUs are allocated through the various channels so that the SU with the highest $\lambda_{i,k}$ (where i represents a sample of 1000 time slots and k the number of the SU node) is selected to the less busy channel. The allocation only occurs one time based on the first sample value of the packet generation rate instead of the average of each node. The throughput is calculated by counting all the packets transmitted by the SU node during the simulation and the average waiting time is the average of time slots needed to send a single packet.

The goal of only considering the first sample value is to demonstrate the algorithm at each stage of operation.

In this case, the three nodes will vary their λ over time although the average of the

different λ is 0.8 packets per second. In the first instance, node 2 is the node with the highest λ, followed by node 1 and finally node 3. The first values of $\lambda_{i,k}$ are:

Table 4.1: The λ_{ik} values of the first algorithm.

i	$\lambda_{i,1}$	$\lambda_{i,2}$	$\lambda_{i,3}$
1	0.84	0.91	0.76
2	0.64	0.86	1.00
3	0.98	0.97	0.84

Next, we present the flowchart of this algorithm in Figure 4.4.

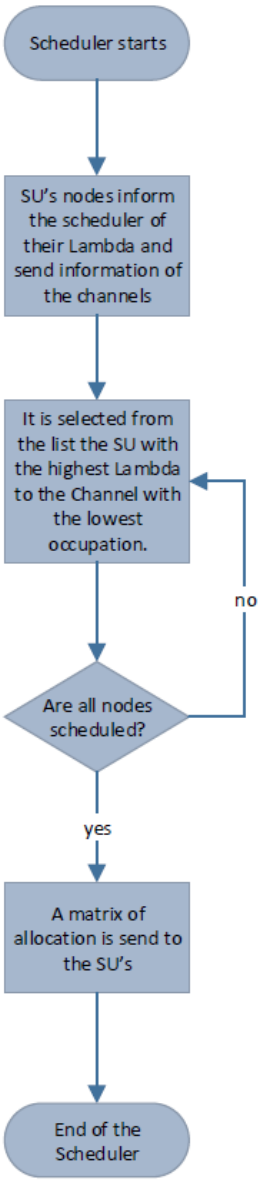


Figure 4.4: First Algorithm scheduler algorithm.

After testing this algorithm, we gathered multiple simulation results. Since we assigned the freer channel to the SU with a larger λ, this was the node that had the highest throughput comparing the other nodes, as it is shown in 4.5. Since node two was able to be placed on the freest channel, it also managed to have a lower average service time than the other nodes as we can see in Figure 4.6.

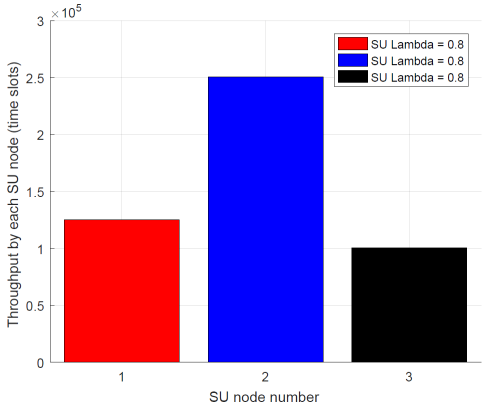


Figure 4.5: Throughput of SU's in the scenario 1.

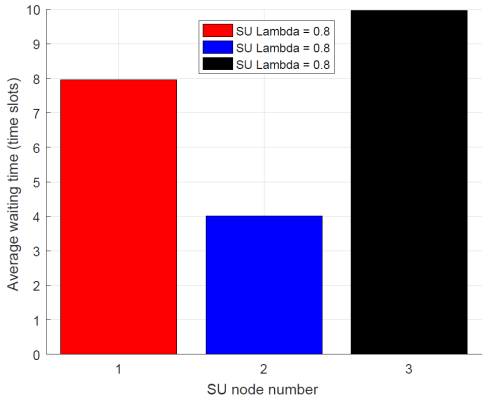


Figure 4.6: Average waiting time of SU's in the scenario 1.

The initial time should not be the choice for the channel selection, because if the three nodes have the same average value of λ, they should not be harmed/benefited due to the first moment in which the algorithm ran. This disadvantage have motivated us to move on to the next channel selection algorithm.

4.4 Second scenario

In this scenario the average values over time of λ belonging to different nodes are equal. The difference between this algorithm and the first one relies on the fact that the scheduling is performed periodically and it takes into account the packet generation rate (λ) sampled over time. This means that the selection matrix between nodes and channels may change

over time depending on the lambda values periodically sampled. In this case, the channels are rescheduled every 1000 time slots, since, the nodes sample their λ value once during this interval.

The flow of the algorithm can be seen in figure 4.7.

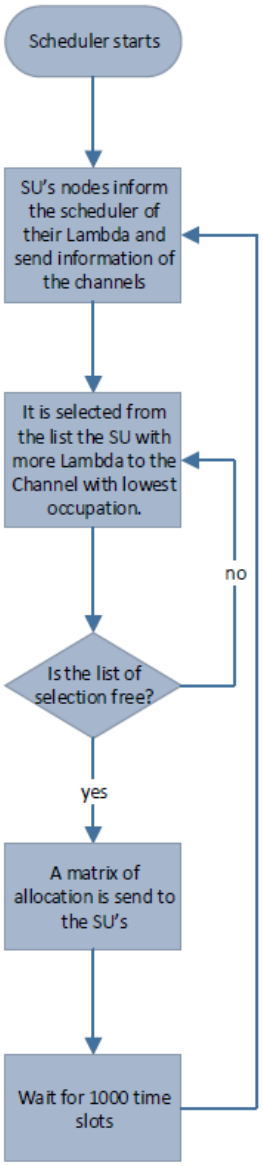


Figure 4.7: Second Algorithm.

The results obtained with the second algorithm are represented in Figures 4.8 and 4.9. The average waiting time was identical between the different nodes. This result was to be expected since, over time, these nodes have the same average λ.

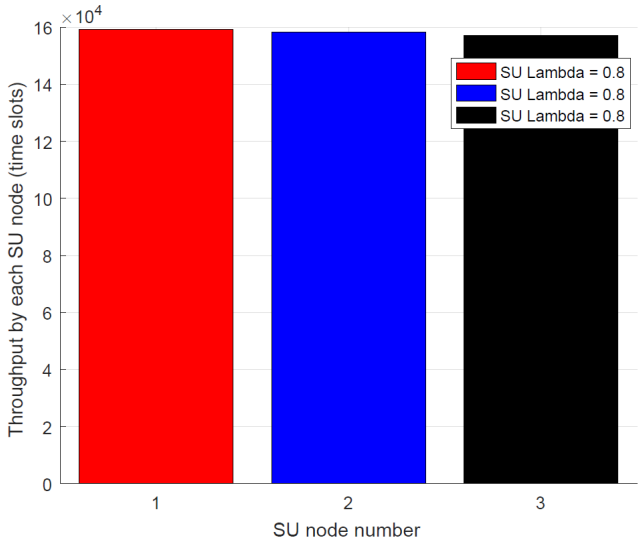


Figure 4.8: Throughput of SU's in the scenario 2.

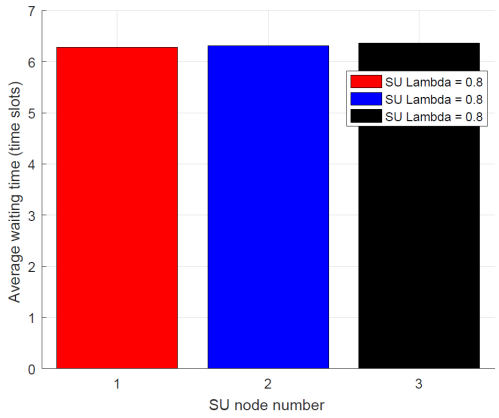


Figure 4.9: Average waiting time of SU's in the scenario 2.

This algorithm has advantages compared to the first one, because it improves the performance when compared to the first one in scenarios where the channel varies over time.

4.5 Third scenario

Another scenario to address is the case where all nodes achieve exactly the same throughput over time, independently of their packet generation rate (λ). In this scenario, the first node has an average λ value of 0.8, the second node 0.4 and finally the third node 0.2. To reach this goal, it was necessary to change the algorithm in order to analyse the accumulated throughput already achieved by each node. The primary goal is to schedule the nodes with the smallest throughput accumulated so far to the channel with higher transmission opportunities.

The algorithm designed for this function is presented below.

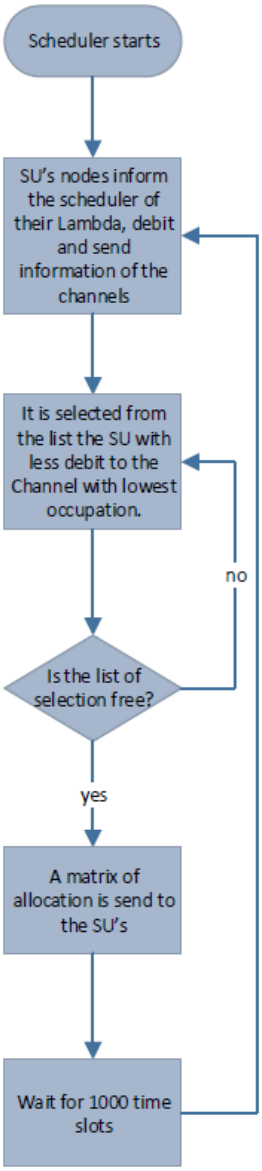


Figure 4.10: Third Algorithm.

The obtained results in Figure 4.11 evaluate the performance of the proposed algorithm. As we can see, even considering different lambdas, the three nodes achieved a similar throughput value. In the case of the waiting time, this was also practically the same between the multiple nodes.

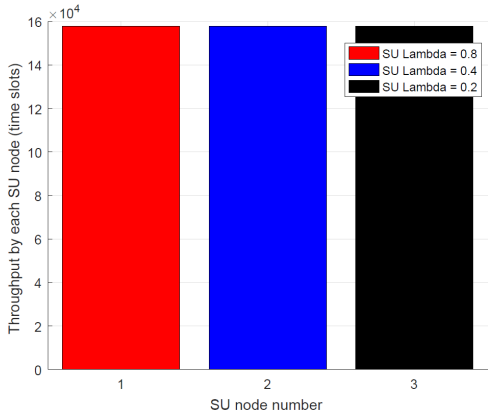


Figure 4.11: Throughput of SU's in the scenario 3.

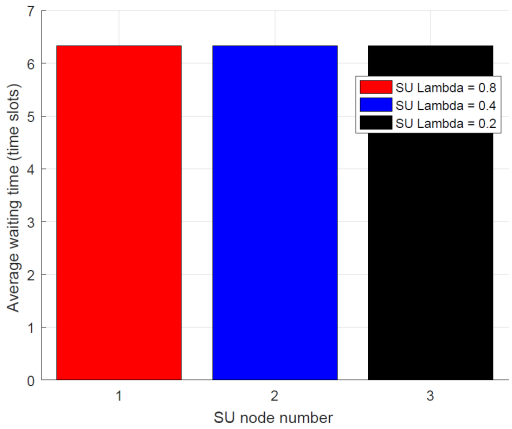


Figure 4.12: Average waiting time of SU's in the scenario 3.

In short, this algorithm is suitable if we want to maximise fairness between the different nodes instead of trying to get the maximum use of the different channels.

4.6 Fourth scenario

In this last scenario, we considered a weighted fairness scheduling policy. The throughput of each node is target to be proportional to its average lambda, that is if we have two nodes in which one of them has one lambda twice as large as the other, the node with higher lambda should double its throughput (when compared to the other node). For this purpose, the algorithm needs to create two lists, one with the nodes below the expected throughput value and other with the nodes that have throughput values higher than expected. The algorithm is represented in Figure 4.13.

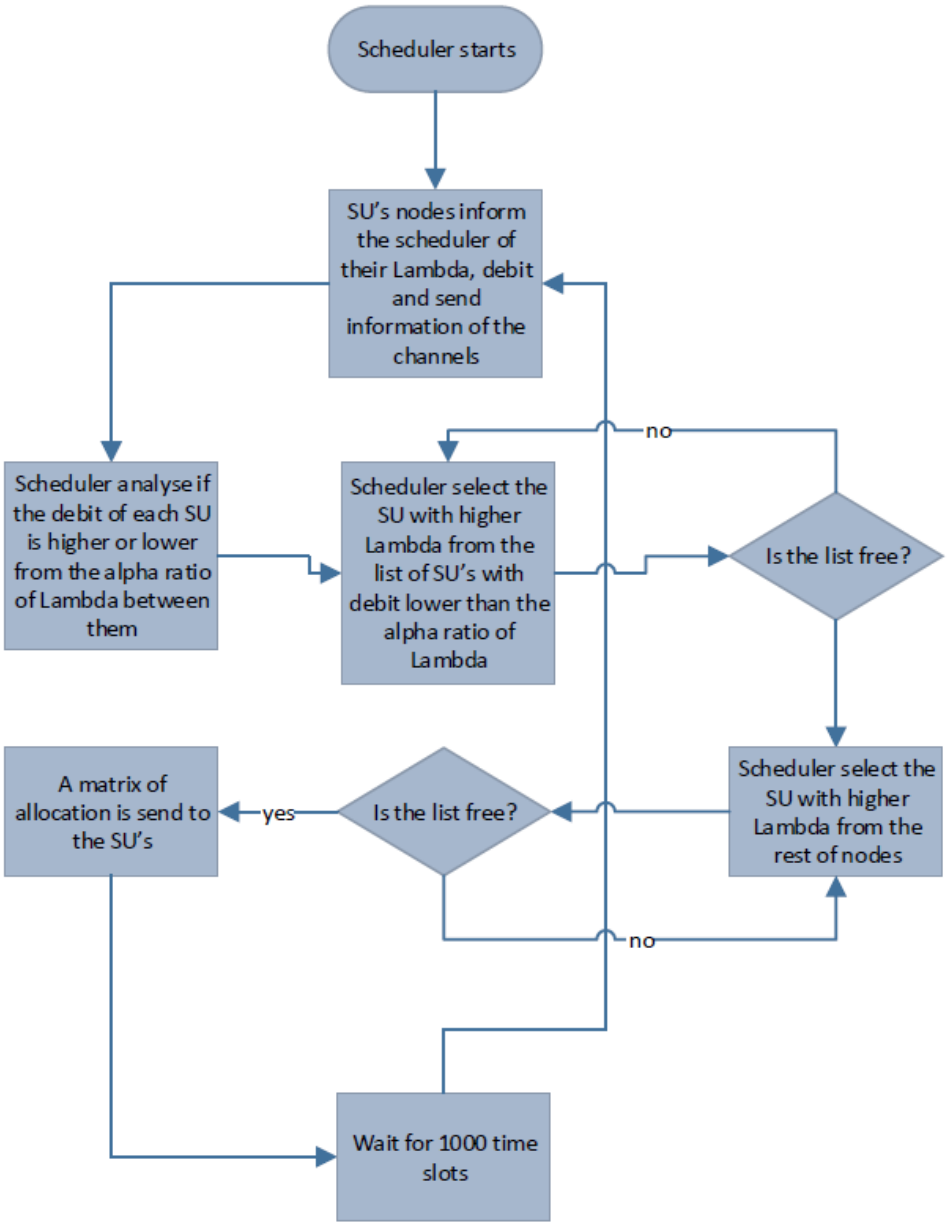


Figure 4.13: Fourth Algorithm.

Figures 4.14 and 4.15 presents the throughput and the average waiting time achieved by the different nodes. Although nodes three and two have a similar rate, the same does not happen between the first node and the second node. The different throughput values achieved by node 1 and 2 is due to their different lambda values. Regarding nodes 2 and 3 we observed that node 3 archives a higher value of throughput than expected because of the polling scheduling policy benefits on the channel idleness.

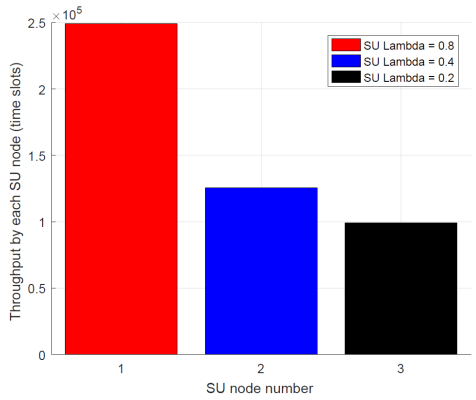


Figure 4.14: Throughput of SU's in the scenario 4.

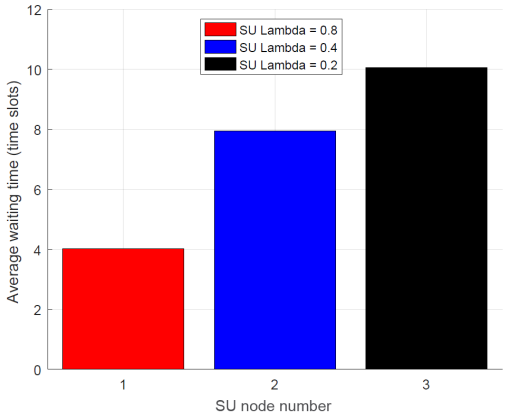


Figure 4.15: Average waiting time of SU's in the scenario 4.

This algorithm is useful for scenarios in which we want the different nodes to have their outputs according to with their needs, not significantly harming the nodes with the lowest λ.

CHAPTER 5

Conclusions and future work

The development of this dissertation allowed us not only to deal with complex themes, such as the work around the spectrum of frequencies but also, the allocation of users through different channels. As we are increasingly dependent on the frequency spectrum for the use of new technologies, the Cognitive Radio is an area that will need to be explored in a near future.

The service time modeling methodology presented in this work will allow the development of increasingly complex algorithms that are able to respond to the challenges posed by a digital society with growing technological needs. The methodology was tested with different simulations and four algorithms were proposed to allocate different users through different channels. Each algorithm corresponds to the need to create an effective response of allocate nodes to the different channels in different scenarios. For each scenario, the priorities for multi-channel packet transmission were delineated.

Taking into account that we created specific algorithms for different traffic situations, an interesting topic to study in future work is the understanding of how to switch between the different algorithms, dynamically.

There is still a long way to go for new developments in this field, such as the creation of algorithms for scenarios where the sensing has a high level of errors. Another interesting study would be the use of artificial intelligence to do the scheduling of the different nodes through multiple channels.

The work developed in this dissertation allows to support the implementation of Cognitive Radiosystems and contributed to better understanding the statistics of such dynamic scenarios.

Bibliography

[1] Z. Caoxie, W. Xinbing, and L. Jun. "Cooperative cognitive radio with priority queueing analysis." In: *IEEE International Conference on Communications* 9.2009 (2009), p. 1. ISSN: 05361486. DOI: 10.1109/ICC.2009.5198858. URL: http://ieeexplore.ieee.org/document/7555343/http://www.scopus.com/inward/record.url?eid=2-s2.0-84872168450{\&}partnerID=40{\&}md5=9aea1eb6570d98812e7e87ddd888d14c.

[2] I. Suliman and J. Lehtomäki. "Queueing analysis of opportunistic access in cognitive radios." In: *2009 2nd International Workshop on Cognitive Radio and Advanced Spectrum Management, CogART 2009* (2009), pp. 153–157. DOI: 10.1109/COGART.2009.5167252.

[3] H. Tran, T. Q. Duong, and H. J. Zepernick. "Average waiting time of packets with different priorities in cognitive radio networks." In: *ISWPC 2010 - IEEE 5th International Symposium on Wireless Pervasive Computing 2010* (2010), pp. 122–127. DOI: 10.1109/ISWPC.2010.5483799.

[4] "Introducing Correlation." In: John Wiley Sons, Ltd, 2011. ISBN: 9781118602218. DOI: 10.1002/9781118602218. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118602218.

[5] X. Li, J. Wang, H. Li, and S. Li. "Delay analysis and optimal access strategy in multichannel dynamic spectrum access system." In: *2012 International Conference on Computing, Networking and Communications, ICNC'12* 2009 (2012), pp. 376–380. DOI: 10.1109/ICCNC.2012.6167447.

[6] M. Luis, R. O. A. Furtado, and L. B. R. Dinis. "Towards a Realistic Primary Users' Behavior in Single Transceiver Cognitive Networks." In: *IEEE Communications Letters* 17 (2013), pp. 309–312.

[7] "Introducing Correlation." In: *Advanced Equity Derivatives*. John Wiley Sons, Ltd, 2014. ISBN: 9781118835364. DOI: 10.1002/9781118835364.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118835364.ch6. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118835364.ch6.

[8] M. Usman, M. S. Khan, H. Vu-Van, and K. Insoo. "Energy-efficient channel handoff for sensor network-assisted cognitive radio network." In: *Sensors (Switzerland)* 15.8 (2015), pp. 18012–18039. ISSN: 14248220. DOI: 10.3390/s150818012.

[9] A. Furtado, L. B. L. Irio R. Oliveira, and R. Dinis. "Spectrum Sensing Performance in Cognitive Radio Networks With Multiple Primary Users." In: *IEEE Transactions on Vehicular Technology* 65 (2016), pp. 1564–1574.

[10] M. Luis, R. Oliveira, R. Dinis, and L. Bernardo. "Characterization of the Opportunistic Service Time in Cognitive Radio Networks." In: *IEEE Transactions on Cognitive Communications and Networking* 7731.c (2016), pp. 1–1. ISSN: 2332-7731. DOI: 10.1109/TCCN.2016.2603994. URL: http://ieeexplore.ieee.org/document/7555343/.

[11] D. Oliveira and R. Oliveira. "Characterization of Energy Availability in RF Energy Harvesting Networks." In: *Mathematical Problems in Engineering* 2016 (2016), p. 9.

[12] R. O. M. Luís, R. Dinis, and L. Bernardo. "RF-Spectrum Opportunities for Cognitive Radio Networks Operating Over GSM Channels." In: *IEEE Transactions on Cognitive Communications and Networking* 3 (2017), pp. 731–739.

ANNEX I

Annex 1

```
// Simulador.m
%INPUT VARIABLES

channel_size = 10^6;
queue_size = 10^6;
PU_packets_size = 1;
SU_packets_size = 1;
meanON = 2;
meanOFF = 18;
PU_Probability_channel = 1/meanON;
PU_Probability_channel_inv = 1/meanOFF;
Lambda = 0.3;

Lambda_s =
    [0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9];
meanON_s = [18,16,14,12,10,8,6,4,2];
meanOFF_s = [2,4,6,8,10,12,14,16,18];

Matriz_final = zeros(162,4);

linha_matriz = 1;

filename = 'C:\Users\Alexandre
    Dias\Desktop\Simulador\Resultados_simulacao_modelo.xlsx'

for n_means_channels = 1:9

    meanON = meanON_s(n_means_channels);
```

```
meanOFF = meanOFF_s(n_means_channels);
PU_Probability_channel = 1/meanON;
PU_Probability_channel_inv = 1/meanOFF;
fprintf('---------------##########----------\nMean ON is %d and Mean OFF is
    %d' ,meanON,meanOFF)

for n_lambdas = 1:18

    Lambda = Lambda_s(n_lambdas);
    MeanWaitingTime = [0,0,0];
    Desvio_padrao = [0,0,0];
    Numero_medidas = [0,0,0];
    Intervalo_confianca = [0,0,0];


    for n_testes = 1:3
        %DONT CHANGE THESE VARIABLES
        %Probability_free = 1 - Probability_occupied;
        iterador_channel_free = 1;
        iterador_channel_occupied = 1;
        flag_occupied = 0;
        flag_idle = 1;
        flag_stop = 0;
        channel_to_occupy = 0;
        channel_to_free = 0;
        sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
        actual_position = 1;
        iterador_to_send = 1;
        iterador_waiting = 1;
        iterador_queue = 1;
        last_added = 1;
        time_waiting = 0;
        acumulador=0;
        iterador_desvio_padrao = 1;

        channel = zeros(1,channel_size,'uint32');
        channel_su = zeros(1, channel_size,'uint32');
        channel_occupied_time = zeros(1,channel_size,'uint32');
        channel_free_time = zeros(1,channel_size,'uint32');
        channel_waiting_times = zeros(1,channel_size,'uint32');
        channel_queue = zeros(1,queue_size,'uint32');
        channel_waiting_each_packet = zeros(1,queue_size,'uint32');

        while actual_position < channel_size
            if flag_occupied == 0
                channel_to_occupy = geornd(PU_Probability_channel);
```

```
            channel_to_occupy = channel_to_occupy + 1;
            channel_occupied_time(iterador_channel_occupied) =
                channel_to_occupy;
            iterador_channel_occupied = iterador_channel_occupied+1;
            for n = 1:channel_to_occupy
                channel(actual_position) = 1;
                actual_position=actual_position+1;
                if actual_position == channel_size
                    n = channel_to_occupy;
                end
                %iterador_channel = iterador_channel + channel_to_occupy;
            end
            if channel_to_occupy ~= 0 || PU_Probability_channel == 1
                flag_free = 0;
                flag_occupied = 1;
            end

        else
            if flag_free == 0
                channel_to_occupy = geornd(PU_Probability_channel_inv);
                channel_to_occupy = channel_to_occupy + 1;
                channel_free_time(iterador_channel_occupied) =
                    channel_to_occupy;
                iterador_channel_free = iterador_channel_free+1;
                for n = 1:channel_to_occupy
                    channel(actual_position) = 0;
                    actual_position=actual_position+1;
                    %iterador_channel = iterador_channel +
                        channel_to_occupy;
                    if actual_position == channel_size
                        n = channel_to_occupy;
                    end
                end
                if channel_to_occupy ~= 0 || PU_Probability_channel == 0
                    flag_occupied = 0;
                    flag_free = 1;
                end
            end
        end

    end

    actual_position = 1;
```

```
while actual_position < channel_size
    if flag_occupied == 0
        channel_to_occupy = geornd(1- Lambda);
        if Lambda == 1
            channel_to_occupy = geornd(0.001);
        end
        channel_to_occupy = channel_to_occupy + 1;
        channel_occupied_time(iterador_channel_occupied) =
            channel_to_occupy;
        iterador_channel_occupied = iterador_channel_occupied+1;
        for n = 1:channel_to_occupy
            channel_su(actual_position) = 1;
            actual_position=actual_position+1;
            if actual_position == channel_size
                n = channel_to_occupy;
            end
            %iterador_channel = iterador_channel + channel_to_occupy;
        end
        if channel_to_occupy ~= 0 || PQE == 1
            flag_free = 0;
            flag_occupied = 1;
        end

    else
        if flag_free == 0
            channel_to_occupy = geornd(Lambda);
            if Lambda == 1
                channel_to_occupy = geornd(0.000000001);
            end
            channel_to_occupy = channel_to_occupy + 1;
            channel_free_time(iterador_channel_occupied) =
                channel_to_occupy;
            iterador_channel_free = iterador_channel_free+1;
            for n = 1:channel_to_occupy
                channel_su(actual_position) = 0;
                actual_position=actual_position+1;
                %iterador_channel = iterador_channel +
                    channel_to_occupy;
                if actual_position == channel_size
                    n = channel_to_occupy;
                end
            end
            if (channel_to_occupy ~= 0) || PQE == 0
                flag_occupied = 0;
                flag_free = 1;
            end
```

```
        end
    end
end


actual_position = 1;
iterador_to_send = 1;


time_waiting = 0;
iterador_queue = 1;
iterador_waiting = 0;
last_queue = 0;

while actual_position < channel_size

    if channel_su(actual_position) == 1
        if channel(actual_position) == 0
            if channel_queue(iterador_queue) == 0
                iterador_waiting = iterador_waiting + 1;
                %time_waiting = time_waiting + 1;
                channel_waiting_times(iterador_waiting) = 1;
                time_waiting = 0;
            else
                last_queue = last_queue + 1;
                channel_queue(last_queue) = 1;
                iterador_queue = iterador_queue + 1;
                iterador_waiting = iterador_waiting + 1;
                time_waiting = time_waiting + 1;
                channel_waiting_times(iterador_waiting) = time_waiting;
                time_waiting = 0;
            end
        else
            last_queue = last_queue + 1;
            channel_queue(last_queue) = 1;
            time_waiting = time_waiting + 1;
        end

    else
        if channel(actual_position) == 0
            if channel_queue(iterador_queue) == 0
                %nao acontece nada
            else
                iterador_queue = iterador_queue + 1;
                iterador_waiting = iterador_waiting + 1;
                time_waiting = time_waiting + 1;
```

```
                    channel_waiting_times(iterador_waiting) = time_waiting;
                    time_waiting = 0;
                end
            else
                if channel_queue(iterador_queue) == 0
                    %nao acontece nada
                else
                    time_waiting = time_waiting + 1;
                end
            end
        end
        actual_position = actual_position + 1;
    end




    MeanWaitingTime(n_testes) =
        mean(channel_waiting_times(1:iterador_waiting));
    Desvio_padrao(n_testes) =
        std(double(channel_waiting_times(1:iterador_waiting)));
    Numero_medidas(n_testes) = iterador_waiting;
    Intervalo_confianca(n_testes) = 1.96 *
        Desvio_padrao/sqrt(Numero_medidas);
end

fprintf('######################\nLambda is %d' ,Lambda)




MeanWaitingTime_final = mean(MeanWaitingTime)
Desvio_padrao_final = mean(Desvio_padrao)
Numero_medidas_final = mean(Numero_medidas)
Intervalo_confianca_final = mean(Intervalo_confianca)

Matriz_final(linha_matriz, 1) = MeanWaitingTime_final;
Matriz_final(linha_matriz, 2) = Desvio_padrao_final;
Matriz_final(linha_matriz, 3) = Numero_medidas_final;
Matriz_final(linha_matriz, 4) = Intervalo_confianca_final;

linha_matriz = linha_matriz + 1;

end
```

```
end

xlswrite(filename,Matriz_final,1,'E3:H164')


%aux = numel(find(channel_queue==1));
%PQE = 1 - (aux/iterador_queue)
```

ANNEX II

Annex 2

```
// Algoritmo.m


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Variables to use %%%%%%%%%%%%%%%%%%%%%%%%%%%%

N_users = 3;
N_channels = 3;
meanON_PUs = [18, 14, 6];
meanOFF_PUs = [2, 6, 14];
iterations = 10^6;
channel_size = 10^6;
queue_size = 10^6;
cenario = 4;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% System Variables %%%%%%%%%%%%%%%%%%%%%%%%%%%%


%PU_Probability_channel = 1/meanON;
PU_Probability_channel = [1/meanON_PUs(1); 1/meanON_PUs(2); 1/meanON_PUs(3)];
```

```
%PU_Probability_channel_inv = 1/meanOFF;
PU_Probability_channel_inv = [1/meanOFF_PUs(1); 1/meanOFF_PUs(2);
    1/meanOFF_PUs(3)];
%SU_lambda = [0.1 0.05 0.075];
PU_Estimation = zeros(N_channels,N_users);
Priority_su = zeros(N_users,2);
Algoritm_decision = zeros(N_users,2);

SU_lambda = zeros (3,10000);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 3 nos in excel %%%%%%%%%%%%%%%%%%%%%%%%%%

No1 = xlsread('No1_3e4cenarios.xlsx');

No2 = xlsread('No2_3e4cenarios.xlsx');

No3 = xlsread('No3_3e4cenarios.xlsx');

No1 = No1./100;

No2 = No2./100;

No3 = No3./100;

SU_lambda (1,:) = No1;

SU_lambda (2,:) = No2;

SU_lambda (3,:) = No3;

Debito = [0 0 0];



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


flag_free = 0;
flag_occupied = 1;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
channel = zeros(3,channel_size,'uint32');
channel_su = zeros(3, channel_size,'uint32');
channel_occupied_time = zeros(3,channel_size,'uint32');
```

```
channel_free_time = zeros(3,channel_size,'uint32');
channel_waiting_times = zeros(3,channel_size,'uint32');
channel_queue = zeros(3,queue_size,'uint32');
channel_waiting_each_packet = zeros(3,queue_size,'uint32');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GERAL PARA OS 4 CENARIOS


iterador_channel_free = [1 1 1];
iterador_channel_occupied = [1 1 1];
for su = 1: N_users

%DONT CHANGE THESE VARIABLES
%Probability_free = 1 - Probability_occupied;
flag_occupied = 0;
flag_idle = 1;
flag_stop = 0;
channel_to_occupy = 0;
channel_to_free = 0;
sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
actual_position = 1;




while actual_position < channel_size
    if flag_occupied == 0
        channel_to_occupy = geornd(PU_Probability_channel(su));
        channel_to_occupy = channel_to_occupy + 1;
        channel_occupied_time(su,iterador_channel_occupied(su)) =
            channel_to_occupy;
        iterador_channel_occupied(su) = iterador_channel_occupied(su)+1;
        for n = 1:channel_to_occupy
            channel(su,actual_position) = 1;
            actual_position=actual_position+1;
            if actual_position == channel_size
                n = channel_to_occupy;
            end
            %iterador_channel = iterador_channel + channel_to_occupy;
        end
        if channel_to_occupy ~= 0 || PU_Probability_channel(su) == 1
            flag_free = 0;
            flag_occupied = 1;
        end
```

```
    else
        if flag_free == 0
            channel_to_occupy = geornd(PU_Probability_channel_inv(channel(su)));
            channel_to_occupy = channel_to_occupy + 1;
            channel_free_time(su,iterador_channel_occupied(su)) =
                channel_to_occupy;
            iterador_channel_free(su) = iterador_channel_free(su)+1;
            for n = 1:channel_to_occupy
                channel(su,actual_position) = 0;
                actual_position=actual_position+1;
                %iterador_channel = iterador_channel + channel_to_occupy;
                if actual_position == channel_size
                    n = channel_to_occupy;
                end
            end
            if channel_to_occupy ~= 0 || PU_Probability_channel(su) == 0
                flag_occupied = 0;
                flag_free = 1;
            end
        end
    end

end

actual_position = 1;

end


for su = 1: N_users
% iterador_channel_free = 1;
% iterador_channel_occupied = 1;
% flag_occupied = 0;
% flag_idle = 1;
% flag_stop = 0;
% channel_to_occupy = 0;
% channel_to_free = 0;
% sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
% actual_position = 1;
% iterador_to_send = 1;
% iterador_waiting = 1;
% iterador_queue = 1;
% last_added = 1;
% time_waiting = 0;
% acumulador=0;
```

```
% iterador_desvio_padrao = 1;



while actual_position < channel_size
    if flag_occupied == 0

        channel_to_occupy = geornd(1-
            SU_lambda(su,idivide(actual_position,int32(100),'ceil')));
        if SU_lambda(su,idivide(actual_position,int32(100),'ceil')) == 1
            channel_to_occupy = geornd(0.01);
        end
        channel_to_occupy = channel_to_occupy + 1;
        channel_occupied_time(su, iterador_channel_occupied(su)) =
            channel_to_occupy;
        iterador_channel_occupied(su) = iterador_channel_occupied(su)+1;
        for n = 1:channel_to_occupy
            channel_su(su,actual_position) = 1;
            actual_position=actual_position+1;
            if actual_position == channel_size
                n = channel_to_occupy;
            end
            %iterador_channel = iterador_channel + channel_to_occupy;
        end
        if channel_to_occupy ~= 0 || PQE == 1
            flag_free = 0;
            flag_occupied = 1;
        end

    else
        if flag_free == 0
            channel_to_occupy =
                geornd(SU_lambda(su,idivide(actual_position,int32(100),'ceil')));
            if SU_lambda(su,idivide(actual_position,int32(100),'ceil')) == 1
                channel_to_occupy = geornd(0.01);
            end
            channel_to_occupy = channel_to_occupy + 1;
            channel_free_time(su,iterador_channel_occupied(su)) =
                channel_to_occupy;
            iterador_channel_free(su) = iterador_channel_free(su)+1;
            for n = 1:channel_to_occupy
                channel_su(su, actual_position) = 0;
                actual_position=actual_position+1;
                %iterador_channel = iterador_channel + channel_to_occupy;
                if actual_position == channel_size
                    n = channel_to_occupy;
```

```
                end
            end
            if (channel_to_occupy ~= 0) || PQE == 0
                flag_occupied = 0;
                flag_free = 1;
            end
        end
    end
end


actual_position = 1;
iterador_to_send = 1;


time_waiting = 0;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CENARIO 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (cenario == 1 )

    Lambda = [SU_lambda(1,idivide(actual_position,int32(100),'ceil'))
        SU_lambda(2,idivide(actual_position,int32(100),'ceil'))
        SU_lambda(3,idivide(actual_position,int32(100),'ceil'))];

[out,idx] = sort(Lambda,'descend');
[out_first, idx_first] = sort(meanON_PUs,'ascend');

Algoritm_decision(idx(1),1) = idx(1);
Algoritm_decision(idx(1),2) = idx_first(1);




for su_number = 2: N_users


    [out_new,idx_new] = sort(meanON_PUs,'ascend');
    for channel_number = 1:N_channels


    y = ismember(idx_new(channel_number),Algoritm_decision(:,2));
```

```
    if( y == 0 )
        Algoritm_decision(idx(su_number),2) = idx_new(channel_number);
        Algoritm_decision(idx(su_number),1) = idx(su_number);
        break
    end

    end

end

Algoritm_decision




for su = 1: N_users

iterador_channel_free = 1;
iterador_channel_occupied = 1;
flag_occupied = 0;
flag_idle = 1;
flag_stop = 0;
channel_to_occupy = 0;
channel_to_free = 0;
sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
actual_position = 1;
iterador_to_send = 1;
iterador_waiting = 1;
iterador_queue = 1;
last_added = 1;
time_waiting = 0;
acumulador=0;
iterador_desvio_padrao = 1;


while actual_position < channel_size

    if channel(Algoritm_decision(su,2),actual_position) == 0 &&
        channel_queue(su,iterador_queue) == 1
        %             channel_waiting_times(iterador_waiting) =
            channel_waiting_each_packet(iterador_queue);
        channel_waiting_times(su,iterador_waiting) = time_waiting;
        iterador_waiting = iterador_waiting + 1;
        Debito(su) = Debito(su)+1;
        time_waiting = 1;
        iterador_queue = iterador_queue + 1;
```

```
    end
    if channel_su(Algoritm_decision(su,1),actual_position) == 1
        if channel(su,actual_position) == 0 && channel_queue(su,iterador_queue)
            == 0
            channel_waiting_times(su,iterador_waiting) = 1;
            iterador_waiting = iterador_waiting + 1;
            Debito(su) = Debito(su)+1;
        end
        if channel(Algoritm_decision(su,2),actual_position) == 1 ||
            channel_queue(su,iterador_queue) == 1
            channel_queue(su,last_added) = 1;
            last_added = last_added + 1;
        end
    end
    if channel(Algoritm_decision(su,2),actual_position) == 1 &&
        channel_queue(su,iterador_queue) == 1
        time_waiting = time_waiting + 1;
    end
    if channel_queue(su,iterador_queue) == 1 && channel_su(su,actual_position)
        == 1
        acumulador = acumulador + 1;
    end


    %     for n = iterador_queue:last_added
    %         channel_waiting_each_packet(n) = channel_waiting_each_packet(n) + 1;
    %     end

    actual_position = actual_position + 1;
end



end

for su = 1: N_users

MeanWaitingTime(su) = mean(channel_waiting_times(su,1:iterador_waiting))
Desvio_padrao(su) = std(double(channel_waiting_times(su,1:iterador_waiting)))

%Intervalo_confianca(su) = 1.96 * Desvio_padrao/sqrt(Debito(su))


end
Debito
Debito_total = sum(Debito)
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CENARIO 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (cenario == 2 )

sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
actual_position = 1;
iterador_waiting = [1 1 1];
iterador_queue = [1 1 1];
last_added = [1 1 1];
time_waiting = [0 0 0];
acumulador=[0 0 0];
contador = 1;


Lambda = [SU_lambda(1,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(2,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(3,idivide(actual_position,int32(100),'ceil'))];

[out,idx] = sort(Lambda,'descend');
[out_first, idx_first] = sort(meanON_PUs,'ascend');

Algoritm_decision(idx(1),1) = idx(1);
Algoritm_decision(idx(1),2) = idx_first(1);



for su_number = 2: N_users


    [out_new,idx_new] = sort(meanON_PUs,'ascend');
    for channel_number = 1:N_channels


    y = ismember(idx_new(channel_number),Algoritm_decision(:,2));
    if( y == 0 )
        Algoritm_decision(idx(su_number),2) = idx_new(channel_number);
        Algoritm_decision(idx(su_number),1) = idx(su_number);
```

```
        break
    end

    end

end

Algoritm_decision


while actual_position < channel_size

    if(contador == 100)
        Algoritm_decision = zeros(N_users,2);
        contador = 0;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Lambda = [SU_lambda(1,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(2,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(3,idivide(actual_position,int32(100),'ceil'))];

[out,idx] = sort(Lambda,'descend');
[out_first, idx_first] = sort(meanON_PUs,'ascend');

Algoritm_decision(idx(1),1) = idx(1);
Algoritm_decision(idx(1),2) = idx_first(1);




for su_number = 2: N_users


    [out_new,idx_new] = sort(meanON_PUs,'ascend');
    for channel_number = 1:N_channels


    y = ismember(idx_new(channel_number),Algoritm_decision(:,2));
    if( y == 0 )
        Algoritm_decision(idx(su_number),2) = idx_new(channel_number);
        Algoritm_decision(idx(su_number),1) = idx(su_number);
        break
    end

    end

end
```

```
Algoritm_decision
    end

for su = 1: N_users


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if channel(Algoritm_decision(su,2),actual_position) == 0 &&
        channel_queue(su,iterador_queue(su)) == 1
        %             channel_waiting_times(iterador_waiting) =
            channel_waiting_each_packet(iterador_queue);
        channel_waiting_times(su,iterador_waiting(su)) = time_waiting(su);
        iterador_waiting(su) = iterador_waiting(su) + 1;
        Debito(su) = Debito(su)+1;
        time_waiting(su) = 1;
        iterador_queue(su) = iterador_queue(su) + 1;
    end
    if channel_su(Algoritm_decision(su,1),actual_position) == 1
        if channel(su,actual_position) == 0 &&
            channel_queue(su,iterador_queue(su)) == 0
            channel_waiting_times(su,iterador_waiting(su)) = 1;
            iterador_waiting(su) = iterador_waiting(su) + 1;
            Debito(su) = Debito(su)+1;
        end
        if channel(Algoritm_decision(su,2),actual_position) == 1 ||
            channel_queue(su,iterador_queue(su)) == 1
            channel_queue(su,last_added(su)) = 1;
            last_added(su) = last_added(su) + 1;
        end
    end
    if channel(Algoritm_decision(su,2),actual_position) == 1 &&
        channel_queue(su,iterador_queue(su)) == 1
        time_waiting(su) = time_waiting(su) + 1;
    end
    if channel_queue(su,iterador_queue(su)) == 1 &&
        channel_su(su,actual_position) == 1
        acumulador(su) = acumulador(su) + 1;
    end


    %     for n = iterador_queue:last_added
    %         channel_waiting_each_packet(n) = channel_waiting_each_packet(n) + 1;
    %     end
```

```
end
actual_position = actual_position + 1;
contador = contador +1;


end

for su = 1: N_users

MeanWaitingTime(su) = mean(channel_waiting_times(su,1:iterador_waiting(su)))
Desvio_padrao(su) = std(double(channel_waiting_times(su,1:iterador_waiting(su))))


%Intervalo_confianca(su) = 1.96 * Desvio_padrao/sqrt(Debito(su))

end
Debito
Debito_total = sum(Debito)

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%% CENARIO 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if (cenario == 3 )

sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
actual_position = 1;
iterador_waiting = [1 1 1];
iterador_queue = [1 1 1];
last_added = [1 1 1];
time_waiting = [0 0 0];
acumulador=[0 0 0];
contador = 1;


Lambda = [SU_lambda(1,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(2,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(3,idivide(actual_position,int32(100),'ceil'))];

[out,idx] = sort(Lambda,'descend');
[out_first, idx_first] = sort(meanON_PUs,'ascend');

Algoritm_decision(idx(1),1) = idx(1);
```

```
Algoritm_decision(idx(1),2) = idx_first(1);



for su_number = 2: N_users


    [out_new,idx_new] = sort(meanON_PUs,'ascend');
    for channel_number = 1:N_channels


    y = ismember(idx_new(channel_number),Algoritm_decision(:,2));
    if( y == 0 )
        Algoritm_decision(idx(su_number),2) = idx_new(channel_number);
        Algoritm_decision(idx(su_number),1) = idx(su_number);
        break
    end

    end

end

Algoritm_decision


while actual_position < channel_size

    if(contador == 100)
        Algoritm_decision = zeros(N_users,2);
        contador = 0;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Lambda = [iterador_waiting(1) iterador_waiting(2) iterador_waiting(3)];

[out,idx] = sort(Lambda,'ascend');
[out_first, idx_first] = sort(meanON_PUs,'ascend');

Algoritm_decision(idx(1),1) = idx(1);
Algoritm_decision(idx(1),2) = idx_first(1);



for su_number = 2: N_users
```

```
    [out_new,idx_new] = sort(meanON_PUs,'ascend');
    for channel_number = 1:N_channels


    y = ismember(idx_new(channel_number),Algoritm_decision(:,2));
    if( y == 0 )
        Algoritm_decision(idx(su_number),2) = idx_new(channel_number);
        Algoritm_decision(idx(su_number),1) = idx(su_number);
        break
    end

    end

end

Algoritm_decision
    end

for su = 1: N_users


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if channel(Algoritm_decision(su,2),actual_position) == 0 &&
        channel_queue(su,iterador_queue(su)) == 1
        %             channel_waiting_times(iterador_waiting) =
            channel_waiting_each_packet(iterador_queue);
        channel_waiting_times(su,iterador_waiting(su)) = time_waiting(su);
        iterador_waiting(su) = iterador_waiting(su) + 1;
        Debito(su) = Debito(su)+1;
        time_waiting(su) = 1;
        iterador_queue(su) = iterador_queue(su) + 1;
    end
    if channel_su(Algoritm_decision(su,1),actual_position) == 1
        if channel(su,actual_position) == 0 &&
            channel_queue(su,iterador_queue(su)) == 0
            channel_waiting_times(su,iterador_waiting(su)) = 1;
            iterador_waiting(su) = iterador_waiting(su) + 1;
            Debito(su) = Debito(su)+1;
        end
        if channel(Algoritm_decision(su,2),actual_position) == 1 ||
            channel_queue(su,iterador_queue(su)) == 1
            channel_queue(su,last_added(su)) = 1;
            last_added(su) = last_added(su) + 1;
        end
    end
```

```
    if channel(Algoritm_decision(su,2),actual_position) == 1 &&
        channel_queue(su,iterador_queue(su)) == 1
        time_waiting(su) = time_waiting(su) + 1;
    end
    if channel_queue(su,iterador_queue(su)) == 1 &&
        channel_su(su,actual_position) == 1
        acumulador(su) = acumulador(su) + 1;
    end


    %     for n = iterador_queue:last_added
    %         channel_waiting_each_packet(n) = channel_waiting_each_packet(n) + 1;
    %     end


end
actual_position = actual_position + 1;
contador = contador +1;









end

for su = 1: N_users

MeanWaitingTime(su) = mean(channel_waiting_times(su,1:iterador_waiting(su)))
Desvio_padrao(su) = std(double(channel_waiting_times(su,1:iterador_waiting(su))))

%Intervalo_confianca(su) = 1.96 * Desvio_padrao/sqrt(Debito(su))

end
Debito
Debito_total = sum(Debito)


end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CENARIO 4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if (cenario == 4 )

sensing_perfect = 1; %1: perfect sensing; 0:imperfect sensing
actual_position = 1;
iterador_waiting = [1 1 1];
iterador_queue = [1 1 1];
last_added = [1 1 1];
time_waiting = [0 0 0];
acumulador=[0 0 0];
contador = 1;
SU_mu = [0.8 0.4 0.2]

alpha = zeros(1,3);



% SU_probability_enter_channel = sum(SU_mu);
%
% alpha(1)=SU_mu(1)/SU_probability_enter_channel;
% alpha(2)=SU_mu(2)/SU_probability_enter_channel;
% alpha(3)=SU_mu(3)/SU_probability_enter_channel;

Lambda = [SU_lambda(1,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(2,idivide(actual_position,int32(100),'ceil'))
    SU_lambda(3,idivide(actual_position,int32(100),'ceil'))];

[out,idx] = sort(Lambda,'descend');
[out_first, idx_first] = sort(meanON_PUs,'ascend');

Algoritm_decision(idx(1),1) = idx(1);
Algoritm_decision(idx(1),2) = idx_first(1);



for su_number = 2: N_users


    [out_new,idx_new] = sort(meanON_PUs,'ascend');
    for channel_number = 1:N_channels


    y = ismember(idx_new(channel_number),Algoritm_decision(:,2));
    if( y == 0 )
```

```
        Algoritm_decision(idx(su_number),2) = idx_new(channel_number);
        Algoritm_decision(idx(su_number),1) = idx(su_number);
        break
    end

    end

end

Algoritm_decision


while actual_position < channel_size

    if(contador == 100)
        Algoritm_decision = zeros(N_users,2);
        contador = 0;
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Lambda = [iterador_waiting(1) iterador_waiting(2) iterador_waiting(3)];


Debito_total = sum(Debito)


[out_su,idx_su] = sort(SU_mu,'descend');
[out_pu,idx_pu] = sort(meanON_PUs,'ascend');

SU_probability_enter_channel = SU_mu(1)+SU_mu(2)+SU_mu(3);
 alpha(1)=SU_mu(1)/SU_probability_enter_channel;
 alpha(2)=SU_mu(2)/SU_probability_enter_channel;
 alpha(3)=SU_mu(3)/SU_probability_enter_channel;



 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(Debito(1) <= (alpha(1)*Debito_total))

    Algoritm_decision(idx(1),1) = idx_su(1);
    Algoritm_decision(idx(1),2) = idx_pu(1);


else

    if(2 < 4)
```

```
        if(Debito(2) <= (alpha(2)*Debito_total))

            Algoritm_decision(idx(2),1) = idx_su(2);
            Algoritm_decision(idx(2),2) = idx_pu(1);

        else

            if (3 < 4)

               Algoritm_decision(idx(3),1) = idx_su(3);
                Algoritm_decision(idx(3),2) = idx_pu(1);

            end

        end

    end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if(Debito(2) <= (alpha(2)*Debito_total))

   Algoritm_decision(idx(2),1) = idx_su(2);
   Algoritm_decision(idx(2),2) = idx_pu(2);

else

    if(Debito(3) <= (alpha(3)*Debito_total))

        Algoritm_decision(idx(3),1) = idx_su(3);
        Algoritm_decision(idx(3),2) = idx_pu(2);


    end


end

for su = 1: N_users

    y = ismember(su,Algoritm_decision(:,1));

    if(y == 0)

        for channel_number = 1:3
```

```
            x = ismember(channel_number,Algoritm_decision(:,2));

            if(x == 0)

            Algoritm_decision(su,2) = channel_number;
            Algoritm_decision(su,1) = su;

            end

        end

    end
end



Algoritm_decision

    end

for su = 1: N_users


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if channel(Algoritm_decision(su,2),actual_position) == 0 &&
        channel_queue(su,iterador_queue(su)) == 1
        %             channel_waiting_times(iterador_waiting) =
            channel_waiting_each_packet(iterador_queue);
        channel_waiting_times(su,iterador_waiting(su)) = time_waiting(su);
        iterador_waiting(su) = iterador_waiting(su) + 1;
        Debito(su) = Debito(su)+1;
        time_waiting(su) = 1;
        iterador_queue(su) = iterador_queue(su) + 1;
    end
    if channel_su(Algoritm_decision(su,1),actual_position) == 1
        if channel(su,actual_position) == 0 &&
            channel_queue(su,iterador_queue(su)) == 0
            channel_waiting_times(su,iterador_waiting(su)) = 1;
            iterador_waiting(su) = iterador_waiting(su) + 1;
            Debito(su) = Debito(su)+1;
        end
        if channel(Algoritm_decision(su,2),actual_position) == 1 ||
            channel_queue(su,iterador_queue(su)) == 1
```

```
            channel_queue(su,last_added(su)) = 1;
            last_added(su) = last_added(su) + 1;
        end
    end
    if channel(Algoritm_decision(su,2),actual_position) == 1 &&
        channel_queue(su,iterador_queue(su)) == 1
        time_waiting(su) = time_waiting(su) + 1;
    end
    if channel_queue(su,iterador_queue(su)) == 1 &&
        channel_su(su,actual_position) == 1
        acumulador(su) = acumulador(su) + 1;
    end


    %     for n = iterador_queue:last_added
    %         channel_waiting_each_packet(n) = channel_waiting_each_packet(n) + 1;
    %     end


end
actual_position = actual_position + 1;
contador = contador +1;


end

for su = 1: N_users

MeanWaitingTime(su) = mean(channel_waiting_times(su,1:iterador_waiting(su)))
Desvio_padrao(su) = std(double(channel_waiting_times(su,1:iterador_waiting(su))))
Debito

%Intervalo_confianca(su) = 1.96 * Desvio_padrao/sqrt(Debito(su))

end

Debito_total = sum(Debito)


end
```